Estimation de la volatilité locale d'actifs financiers par une méthode d'inversion numérique

J. Frederic Bonnans, Jean-Marc Cognet, Sophie Volle

INRIA

INSTITUT NATIONAL DE RECHERCHE EN INFORMATIQUE ET EN AUTOMATIQUE

Estimation de la volatilité locale d'actifs financiers par une méthode d'inversion numérique

J. Frédéric Bonnans — Jean-Marc Cognet — Sophie Volle

N° 4648

Novembre 2002

THÈME 4

ISSN 0249-6399 ISRN INRIA/RR--4648--FR

INRIA
ROCQUENCOURT

Estimation de la volatilité locale d'actifs financiers par une méthode d'inversion numérique

J. Frédéric Bonnans* , Jean-Marc Cognet† , Sophie Volle‡

Thème 4 — Simulation et optimisation
de systèmes complexes
Projets Mathfi et Sydoco

Rapport de recherche n° 4648 — Novembre 2002 — 60 pages

Résumé : Nous nous intéressons à un problème de calibration rencontré en finance qui consiste à identifier la volatilité locale à partir de prix d'options observés sur le marché. Nous formulons le problème sous la forme d'un problème inverse. Le problème direct (pricing), qui consiste à calculer les prix d'options en résolvant l'équation aux dérivées partielles de Dupire, est résolu par différences finies. Le problème inverse (calibration) revient ensuite à résoudre un problème d'optimisation dans lequel il s'agit de déterminer la volatilité optimale telle que les prix calculés avec cette volatilité soit le plus proche possible des prix observés sur le marché. Ce problème inverse mal posé est régularisé en paramétrant la volatilité par des splines bicubiques et en adoptant une approche multiéchelle. De plus, nous utilisons l'algorithme de Quasi-Newton (avec ou sans bornes) pour résoudre le problème d'optimisation. Notre algorithme de calibration est testé sur des donnés synthétiques (simulées par notre modèle) et sur des données réelles du marché.

Mots-clés : finance, calibration, EDP de Dupire, différences finies, problème inverse, optimisation

* Projet Sydoco

† EDF Recherche et Développement, 1 avenue du Général de Gaulle, BP 408, 92141 Clamart. Travail effectué dans le cadre d'un Post-Doctorat Industriel au sein du projet Mathfi

‡ Raise Partner, 3 Chemin de ronde, 38000 Grenoble. Travail effectué dans le cadre d'un poste d'accueil des projets Mathfi et Sydoco

Unité de recherche INRIA Rocquencourt
Domaine de Voluceau, Rocquencourt, BP 105, 78153 Le Chesnay Cedex (France)
Téléphone : +33 1 39 63 55 11 —Télécopie : +33 1 39 63 53 30

Estimation of the local volatility of financial assets by a numerical inversion method

Abstract: We consider a problem arising in finance that consists in identifying the local volatility from option prices provided by the market. The problem is formulated as an inverse problem. The direct problem (pricing), which consists in computing option prices by solving Dupire's partial differential equation, is solved by finite differences. Then, the inverse problem (calibration) consists in solving an optimization problem where we have to determine the optimal volatility such that the prices computed with this volatility are as close as possible to the prices given by the market. This ill-posed inverse problem is regularized by parameterizing the volatility using bicubic splines and by employing a multiscale approach. Moreover, we use the Quasi-Newton algorithm (with or without bounds) to solve the optimization problem. Our algorithm of calibration is tested on synthetic data (simulated with our model) and on real data provided by the market.

Key-words: finance, calibration, Dupire's PDE, finite differences, inverse problem, optimization

Table des matières

1 Introduction

Le modèle de Black-Scholes suppose que l'évolution du prix de l'actif est régi par l'équation différentielle stochastique suivante (cf. [3]) :

$$dS_t = S_t(\mu dt + \sigma dB_t),$$

où la volatilité σ du sous-jacent est supposée constante et où B_t est un mouvement brownien. A partir de ce modèle de diffusion de l'actif, on peut déterminer le prix des options sur cet actif. Inversement, la volatilité σ étant le seul paramètre non observable de ce modèle, on s'intéresse au calcul de σ à partir du prix d'une option (K, T) observé sur le marché. La volatilité ainsi obtenue est appelée volatilité implicite (notée $\sigma_{BS}(K, T)$).

Lorsque l'on confronte ce modèle aux données réelles, on réalise ses limites : la volatilité implicite $\sigma_{BS}(K, T)$ est très sensible aux variations du strike K et de la maturité T. Or, la volatilité d'un actif ne peut pas dépendre de l'option que l'on considère. Ce phénomène est connu sous le nom de "smile" (cf. [10]). Il est important d'estimer la volatilité le plus justement possible pour ensuite pouvoir évaluer des options plus complexes. Il faut donc trouver une autre représentation de σ.

De nombreux articles se sont intéressés à ce problème ([12], [8], [11], [1]), en proposant un nouveau processus de diffusion pour le sous-jacent (cf. par exemple [10], [12]) :

$$dS_t = S_t(\mu dt + \sigma(S_t, t) dB_t),$$

où la volatilité $\sigma(S_t, t)$ dépend maintenant du cours du sous-jacent et du temps. Dans le modèle de base de Black-Scholes (avec σ constant), on peut facilement déduire σ_{BS} du prix de l'option. Dans le cas de la volatilité locale, déterminer la nappe $\sigma(S_t, t)$ est beaucoup plus délicat. Cela revient en fait à résoudre un problème inverse sous-déterminé et mal posé.

Dans la méthode présentée ici, la fonction de volatilité apparaît comme coefficient d'une EDP (EDP de Dupire) vérifiée par le prix de l'option. On cherche à déterminer ce coefficient à partir de l'information dont on dispose concernant le prix gouverné par l'EDP. En principe, ce problème inverse peut être résolu si l'on a assez d'information sur les prix d'options, mais dans la pratique :

- le problème inverse est sous-déterminé car le marché ne fournit qu'une quantité limitée d'information,
- le problème est mal posé car la volatilité ne dépendra pas continûment des données du marché.

Pour remédier à ces obstacles, nous utilisons une technique de régularisation qui permet de faire un compromis entre la précision (les données simulées avec notre modèle sont proches des données du marché) et la stabilité en recherchant une volatilité suffisamment régulière.

Dans un premier temps, nous ajoutons un terme de régularisation $\lambda\|\sigma\|^2$ à la fonction coût mesurant l'écart entre les données simulées et les données du marché. La difficulté revient à déterminer le coefficient λ le mieux adapté. Après avoir montré des premiers tests de calibration sur données synthétiques, nous proposons une autre méthode pour régulariser le problème : l'approche multiéchelle qui est décrite à la section 4.2.

Plusieurs articles ont déjà attaqué ce problème avec le même type d'approche "problème inverse" ([12], [11], [8]). La méthode proposée ici diffère cependant en plusieurs points, dont nous discuterons l'opportunité :

- la formulation du problème direct (EDP de Dupire et non de Black-Scholes),
- le paramétrage de σ (spline bicubique non contraint),
- la méthode d'optimisation (Quasi-Newton avec calcul exact du gradient, aux erreurs d'arrondis près).

Dans un premier temps, nous présentons une méthode de résolution du problème direct de "pricing" (EDP de Dupire). S'ensuit une discussion concernant la régularisation du problème et le choix de la méthode d'optimisation à utiliser. Notre choix se porte sur une approche multiéchelle et une méthode de Quasi-Newton avec bornes, couplées avec un paramétrage de la volatilité par des splines bicubiques. Nous validons notre méthode de calibration sur deux jeux de données synthétiques et sur un jeu de données réelles observées sur le marché.

2 Problème direct : résolution numérique de l'EDP parabolique de Dupire

2.1 Hypothèses et données

On considère une option européenne de type call (on peut de manière similaire traiter une option de type put). Soient :

- $t_0 \geq 0$ l'origine du temps,
- S_0 le prix de l'actif sous-jacent à l'instant t_0,
- K le prix d'exercice,
- T la maturité,
- $\sigma(S,t)$ la volatilité de l'actif sous-jacent de prix S à la date t,
- $V(K,T;S_0,t_0,\sigma)$, que l'on notera $V(K,T)$, le prix de l'option de prix d'exercice K, de maturité T, à l'instant t_0 lorsque l'actif sous-jacent vaut S_0 et la volatilité vaut σ [1],

1. Contrairement à l'équation de Black-Scholes, S_0 et t_0 sont les paramètres du problèmes tandis que K et T sont les variables.

- r le taux d'intérêt constant sans risque,
- q le rendement continu constant de l'actif,
- le prix de l'actif sous-jacent est gouverné par l'équation différentielle suivante (voir par exemple [3]) :

$$\frac{dS}{S} = (r-q)dt + \sigma(S,t)dW \tag{1}$$

où W est un mouvement brownien.

2.2 Formulation de l'EDP de Dupire à partir de l'équation de Fokker-Planck

Le prix de l'actif S vérifie l'équation différentielle :

$$dS = S(r-q)dt + S\sigma(S,t)dW.$$

L'équation de Fokker-Planck est donc (cf. [10]) :

$$\partial_t p(S,t) = -\partial_S((r-q)Sp(S,t)) + \partial_{SS}(\frac{\sigma(S,t)^2}{2}S^2 p(S,t)), \tag{2}$$

où $p(.,t)$ est la densité du prix du sous-jacent à la date t. Par définition, le prix de l'option de prix d'exercice K, de maturité T, à l'instant t_0 est :

$$V(K,T) = e^{-r(T-t_0)} \int_0^\infty (S-K)_+ p(S,T)dS.$$

On dérive les deux côtés par rapport à T en tenant compte de (2) :

$$\begin{aligned}\partial_t V(K,T) &= -rV(K,T) + e^{-r(T-t_0)} \int_0^\infty (S-K)_+ \Big[-\partial_S((r-q)Sp(S,T)) \\ &\quad +\partial_{SS}(\tfrac{1}{2}\sigma(S,T)^2 S^2 p(S,T))\Big] dS.\end{aligned}$$

Par une intégration par partie, on obtient :

$$\begin{aligned}
\partial_t V(K,T) &= -rV(K,T) - e^{-r(T-t_0)} \int_K^\infty \partial_S(\tfrac{1}{2}\sigma^2(S,T)S^2 p(S,T))dS \\
&\quad + e^{-r(T-t_0)} \int_0^\infty (r-q)(S-K)_+ p(S,T)dS \\
&\quad + K(r-q)e^{-r(T-t_0)} \int_K^\infty p(S,T)dS \\
&= -rV(K,T) + e^{-r(T-t_0)} \tfrac{1}{2}\sigma^2(K,T)K^2 p(K,T) \\
&\quad + (r-q)e^{-r(T-t_0)} \int_0^\infty (S-K)_+ p(S,T)dS \\
&\quad + K(r-q)e^{-r(T-t_0)} \int_K^\infty p(S,T)dS.
\end{aligned}$$

Puis, grâce aux égalités suivantes, après calculs :

$$\begin{aligned}
V(K,T) &= e^{-r(T-t_0)} \int_0^\infty (S-K)_+ p(S,T)dS, \\
\partial_K V(K,T) &= -e^{-r(T-t_0)} \int_K^\infty p(S,T)dS, \\
\partial_{KK} V(K,T) &= e^{-r(T-t_0)} p(K,T).
\end{aligned}$$

On en déduit que :

$$\begin{aligned}
\partial_t V(K,T) &= -rV(K,T) + \frac{1}{2}\sigma^2(K,T)K^2 \partial_{KK} V(K,T) \\
&\quad + (r-q)V(K,T) - (r-q)K\partial_K V(K,T),
\end{aligned}$$

d'où :

$$\frac{\partial V}{\partial T}(K,T) = -qV(K,T) - (r-q)K\frac{\partial V}{\partial K}(K,T) + \frac{1}{2}\sigma^2(K,T)K^2\frac{\partial^2 V}{\partial K^2}(K,T)$$

sous les conditions aux limites suivantes :

$$\begin{cases}
V(K,t_0) = \max(S_0 - K, 0) & \text{pour } 0 \le K, \\
\lim_{K\to 0} V(K,T) = S_0 e^{-q(T-t_0)} & \text{pour } t_0 \le T, \\
\lim_{K\to+\infty} V(K,T) = 0 & \text{pour } t_0 \le T.
\end{cases}$$

2.3 Changement de variable logarithmique

On choisit de faire un changement de variable logarithmique pour obtenir une condition de stabilité moins contraignante sur les pas de temps et d'espace. En posant :

$$\begin{aligned}
y &= \ln(K), \\
U(y,T) &= V(K,T), \\
\hat{\sigma}(y,T) &= \sigma(K,T),
\end{aligned}$$

on obtient l'EDP suivante :

$$\frac{\partial U}{\partial T}(y,T) = -qU(y,T) - (r-q+\frac{1}{2}\hat{\sigma}^2(y,T))\frac{\partial U}{\partial y}(y,T) + \frac{1}{2}\hat{\sigma}^2(y,T)\frac{\partial^2 U}{\partial y^2}(y,T),$$

avec les conditions aux limites suivantes :

$$\begin{cases} U(y,t_0) = \max(S_0 - e^y, 0) & \text{pour tout } y \text{ réel,} \\ \lim_{y\to-\infty} U(y,T) = S_0 e^{-q(T-t_0)} & \text{pour } t_0 \leq T, \\ \lim_{y\to+\infty} U(y,T) = 0 & \text{pour } t_0 \leq T. \end{cases}$$

Nous résolvons numériquement cette EDP de Dupire par différences finies. La méthode de discrétisation que nous choisissons s'inspire de celle présentée dans [13].

2.4 Discrétisation uniforme

On définit les opérateurs A et $\tilde{A}$ de la manière suivante :

$$\begin{aligned} (AU)(y,T) &= -\frac{1}{2}\hat{\sigma}^2(y,T)\frac{\partial^2 U}{\partial y^2}(y,T) + (r-q+\frac{1}{2}\hat{\sigma}^2(y,T))\frac{\partial U}{\partial y}(y,T), \\ (\tilde{A}U)(y,T) &= (AU)(y,T) + qU(y,T). \end{aligned}$$

Remarque 1 *Le signe du coefficient $(r-q+\frac{1}{2}\hat{\sigma}^2(y,T))$ de la dérivée au premier ordre peut varier. En effet, on a logiquement $r-q<0$ car le rendement q de l'actif risqué doit être supérieur au rendement de l'actif sans risque. On peut penser que $r-q$ sera de l'ordre de 10^{-2}. $\hat{\sigma}^2(y,T)$ sera lui aussi de l'ordre de 10^{-2} puisque la volatilité est de l'ordre de 10^{-1}. Cette incertitude concernant le signe nous pousse à utiliser une approximation centrée des dérivées lors de la discrétisation de l'EDP.*

2.4.1 Discrétisation en espace

En se restreignant à $y \in [y_{min}; y_{max}]$ et $T \in [t_0; T_{max}]$, on obtient l'équation semi-discrétisée suivante :

$$(E)\begin{cases} \frac{\partial U}{\partial T}(y,T) + \tilde{A}U(y,T) = 0 & \text{dans }]y_{min}; y_{max}[\times[t_0; T_{max}], \\ U(y_{min},T) = S_0 e^{-q(T-t_0)} & \text{si } T \in [t_0; T_{max}], \\ U(y_{max},T) = 0 & \text{si } T \in [t_0; T_{max}], \\ U(y,t_0) = f(y) = \max(S_0 - e^y, 0) & \text{pour } y \in]y_{min}; y_{max}[. \end{cases}$$

Soit $h = (y_{max} - y_{min})/N$ le pas d'espace. On pose, pour i allant de 0 à N :

$$\begin{aligned} y_i &= y_{min} + ih, \\ f_i &= f(y_i). \end{aligned}$$

Soit $u(T) = (U(y_i, T))_{1 \leq i \leq N-1} \in I\!R^{N-1}$. A chaque instant, on discrétise $\tilde{A}$ par un opérateur discret $\tilde{A}_T : u(T) \in R^{N-1} \rightarrow \tilde{A}_T u(T) \in I\!R^{N-1}$. Pour cela, on remplace

$$\begin{aligned} \frac{\partial U}{\partial y}(y_i, T) \quad &\text{par} \quad \frac{u_{i+1}(T) - u_{i-1}(T)}{2h}, \\ \frac{\partial^2 U}{\partial y^2}(y_i, T) \quad &\text{par} \quad \frac{u_{i+1}(T) - 2u_i(T) + u_{i-1}(T)}{h^2}. \end{aligned}$$

Posons

$$\begin{aligned} \alpha_{i,T} &= -\frac{\hat{\sigma}^2(y_i, T)}{2h^2} - \frac{1}{2h}(r - q + \frac{\hat{\sigma}^2(y_i, T)}{2}), \\ \beta_{i,T} &= \frac{\hat{\sigma}^2(y_i, T)}{h^2} + q, \\ \gamma_{i,T} &= -\frac{\hat{\sigma}^2(y_i, T)}{2h^2} + \frac{1}{2h}(r - q + \frac{\hat{\sigma}^2(y_i, T)}{2}). \end{aligned}$$

On a alors, pour $i \in \{1, ..., N-1\}$ et pour tout T :

$$(\tilde{A}_T u(T))_i \quad = \quad \alpha_{i,T} u_{i-1}(T) + \beta_{i,T} u_i(T) + \gamma_{i,T} u_{i+1}(T).$$

Conditions aux limites Pour $i = 0$, on a pour tout T une condition de type Dirichlet $u_0(T) = S_0 e^{-q(T-t_0)}$. Par conséquent,

$$(\tilde{A}_T u(T))_1 \quad = \quad \alpha_{1,T} S_0 e^{-q(T-t_0)} + \beta_{1,T} u_1(T) + \gamma_{1,T} u_2(T).$$

Pour $i = N$ ($y = y_{max}$), on a pour tout T la condition de Dirichlet $u_N(T) = 0$, donc

$$(\tilde{A}_T u(T))_{N-1} \quad = \quad \alpha_{N-1,T} u_{N-2}(T) + \beta_{N-1,T} u_{N-1}(T).$$

En particulier, pour $T = t_0$:

$$\left\{ \begin{array}{lcl} f_0 & = & S_0 e^{-q(T-t_0)}, \\ f_N & = & 0. \end{array} \right.$$

Construction de l'opérateur On définit l'opérateur intermédiaire $\hat{A}_T$ de $I\!R^{N-1}$ représenté par la matrice suivante :

$$\left((\hat{A}_T)_{ij} \right)_{1 \leq i,j \leq N} \quad = \quad \begin{pmatrix} \beta_{1,T} & \gamma_{1,T} & 0 & \cdots & 0 \\ \alpha_{2,T} & \beta_{2,T} & \gamma_{2,T} & \ddots & \vdots \\ 0 & \ddots & \ddots & \ddots & 0 \\ \vdots & \cdots & \alpha_{N-2,T} & \beta_{N-2,T} & \gamma_{N-2,T} \\ 0 & 0 & \cdots & \alpha_{N-1,T} & \beta_{N-1,T} \end{pmatrix}$$

A cause de la condition de Dirichlet en $i = 0$, on a alors :

$$\begin{aligned}(\tilde{A}_T u(T))_1 &= \alpha_{1,T} S_0 e^{-q(T-t_0)} + (\hat{A}_T u(T))_1, \\ (\tilde{A}_T u(T))_i &= (\hat{A}_T u(T))_i \quad \forall i \in 2, ..., N-1.\end{aligned}$$

EDP discrétisée en espace Cette discrétisation en espace permet de ramener l'EDP (E) à l'EDO (E_h) :

$$(E^h) \begin{cases} \dfrac{\partial u}{\partial T}(T) + \tilde{A}_T u(T) = 0 \quad \text{si } T \in [t_0; T_{max}] \\ u(t_0) = f \end{cases}$$

soit

$$(E^h) \begin{cases} \dfrac{\partial u}{\partial T}(T) + \left(\hat{A}_T u(T) + \alpha_{1,T} S_0 e^{-q(T-t_0)} e_1 \right) = 0 \quad \text{si } T \in [t_0; T_{max}] \\ u(t_0) = f \end{cases}$$

où $f = (f(y_i))_{1 \le i \le N-1}$ et $e_1 = \begin{pmatrix} 1 \\ 0 \\ \vdots \\ 0 \end{pmatrix}$.

2.4.2 Discrétisation en temps par les θ-schémas

On utilise ici une généralisation de la discrétisation par les θ-schémas proposée par [13]. Soient $\theta \in [0; 1]$ fixé et k un pas de temps tel que $T_{max} - t_0 = Mk$. On approxime la solution u de (E^h) aux instants $t_0 + nk$ par les u^n solutions de :

$$(E^{h,k}) \begin{cases} u^0 = f \\ \dfrac{u^{n+1} - u^n}{k} + \theta \tilde{A}^n u^n + (1-\theta) \tilde{A}^{n+1} u^{n+1} = 0 \quad \text{pour} \quad 0 \le n \le M-1 \end{cases}$$

soit

$$(E^{h,k}) \begin{cases} u^0 = f \\ \dfrac{u^{n+1} - u^n}{k} + \theta(\hat{A}^n u^n + \alpha_{1,T_n} S_0 e^{-q(T_n - t_0)} e_1) \\ +(1-\theta)(\hat{A}^{n+1} u^{n+1} + \alpha_{1,T_{n+1}} S_0 e^{-q(T_{n+1} - t_0)} e_1) = 0 \quad \text{pour} \quad 0 \le n \le M-1 \end{cases}$$

où l'opérateur $\hat{A}^n$ est en fait l'opérateur $\hat{A}_{(t_0+nk)}$.

On obtient différents types de schémas selon la valeur de θ :

- si $\theta = 1$, le schéma est explicite,
- si $0 \le \theta < 1$, le schéma est implicite.

2.4.3 Résolution

On doit résoudre à chaque étape un système linéaire

$$H^n u^{n+1} = b^n$$

où

$$\begin{aligned} b^n &= (I - \theta k \hat{A}^n) u^n - S_0 k \left(\theta \alpha_{1,T_n} e^{-q(T_n - t_0)} + (1-\theta) \alpha_{1,T_{n+1}} e^{-q(T_{n+1} - t_0)} \right) e_1, \\ H^n &= I + (1-\theta) k \hat{A}^{n+1}. \end{aligned}$$

avec H^n de taille $(N-1, N-1)$ et tridiagonale pour tout n. Il suffit alors de trianguláriser ce système à chaque pas de temps par la méthode du pivot et de le résoudre.

2.4.4 Tests numériques

Pour le cas où la volatilité est constante, on dispose de la formule explicite de Black-Scholes :

$$V(K, T; S_0, t_0) = S_0 e^{-q(T-t_0)} N(d_1) - K e^{-r(T-t_0)} N(d_2)$$

avec

$$\begin{aligned} d_1 &= \frac{\ln(S_0/K) + (r - q + \sigma^2/2)(T - t_0)}{\sigma \sqrt{T - t_0}}, \\ d_2 &= d_1 - \sigma \sqrt{T - t_0}. \end{aligned}$$

et $N(x)$ est la fonction de répartition d'une variable normale de moyenne nulle et de variance 1 :

$$N(x) = \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{x} e^{-\frac{t^2}{2}} dt.$$

Si la volatilité dépend du temps seulement, on a plus généralement la formule suivante (voir par exemple [13]) :

$$V(K, T; S_0, t_0) = S_0 e^{-q(T-t_0)} N(d_1) - K e^{-r(T-t_0)} N(d_2),$$

avec

$$\begin{aligned} d_1 &= \frac{\ln(S_0/K) + (r - q + \Sigma^2_{t_0,T}/2)(T - t_0)}{\Sigma_{t_0,T} \sqrt{T - t_0}}, \\ d_2 &= d_1 - \Sigma_{t_0,T} \sqrt{T - t_0} \end{aligned}$$

et

$$\Sigma^2_{t_0,T} = \frac{1}{T - t_0} \int_{t_0}^{T} \sigma^2(t) dt.$$

Testons en premier lieu le cas σ = constante, par exemple $\sigma = 0.2$. Avec $N = 400$, $M = 100$, $\theta = 0.5$, $r = 0.05$, $q = 0.02$ et $S_0 = 100$, on compare les résultats obtenus par la méthode développée précédemment (figure 1 à gauche) à ceux obtenus avec la formule explicite de Black-Scholes en mesurant la différence séparant les deux (figure 1 à droite). On voit sur cet exemple que l'erreur maximale est concentrée autour de S_0. Pour pallier à ce problème, nous proposons une discrétisation en espace non uniforme (section 2.5) de sorte qu'il y ait plus de valeurs discrètes autour de S_0.

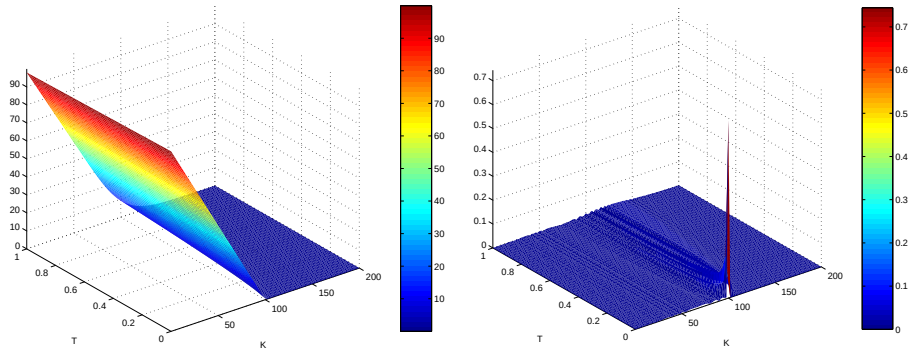


FIG. 1: $\sigma = 0.2$. *Discrétisation uniforme en espace. A gauche : Nappe de prix obtenue par l'EDP de Dupire. A droite : Différences de prix entre Dupire et Black-Scholes.*

2.5 Discrétisation non uniforme

On a vu que la résolution de l'EDP donnait des résultats moins bons autour de S_0, c'est à dire dans la région qui nous intéresse le plus à priori. Pour remédier à cela, on peut modifier la discrétisation de manière à ce que le pas d'espace soit plus petit autour de S_0, quitte à être plus grand aux extrémités. Dans le cas où $y_{min} = -y_{max}$, une solution pour construire une telle grille de prix est la suivante :

- construire une grille uniforme sur $[-y_{max}; +y_{max}]$,
- prendre l'image de cette grille par l'inverse de la fonction

$$x \to y_{max} * \tanh(x - y_0),$$

où $y_0 = \ln(S_0)$.

En effet, on voit sur le graphe de cette fonction (figure 2) que les images inverses de cette fonction sont plus denses autour de y_0 et se raréfient aux extrémités. La fonction inverse de $x \to y_{max} * \tanh(x - y_0)$ est la fonction définie par

$$g(y) = \frac{1}{2}\log((y_{max} + y)/(y_{max} - y)) + y_0.$$

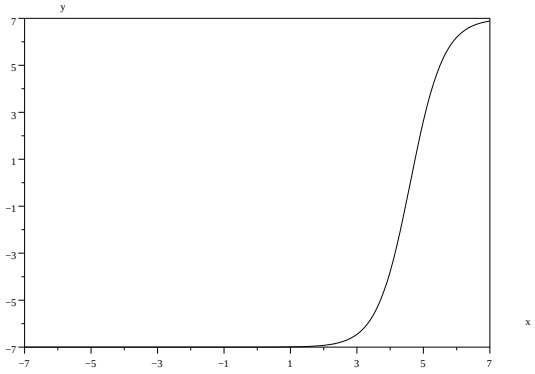


FIG. 2: *Graphe de la fonction* $x \to y_{max} * \tanh(x - y_0)$ *pour* $y_{max} = 7$ *et* $y_0 = \log(100) = 4.60$.

2.5.1 Discrétisation et construction de l'opérateur

Comme le pas d'espace h n'est plus constant, les formules de discrétisation sont différentes. En se restreignant à $y \in [-y_{max}; y_{max}]$ et $T \in [t_0; T_{max}]$, on obtient l'équation suivante :

$$(E) \begin{cases} \frac{\partial U}{\partial T}(y,T) + \tilde{A}U(y,T) = 0 & \text{dans }]-y_{max}; y_{max}[\times[t_0; T_{max}], \\ U(-y_{max},T) = S_0 e^{-q(T-t_0)} & \text{si } T \in [t_0; T_{max}], \\ U(y_{max},T) = 0 & \text{si } T \in [t_0; T_{max}], \\ U(y,t_0) = f(y) = max(S_0 - e^y, 0) & \text{pour } y \in]y_{max}; y_{max}[. \end{cases}$$

Posons $h = 2y_{max}/N$. On définit une discrétisation de l'espace des prix $[-y_{max}; +y_{max}]$ de la manière suivante :

- Soit $i_0 \in 0, ..., N$ le plus petit indice tel que $g(-y_{max} + i_0 \cdot h) > -y_{max}$. Entre $-y_{max}$ et $g(-y_{max} + i_0 \cdot h)$, l'espace est discrétisé de manière uniforme à l'aide de i_0 points $y_0, ..., y_{i_0 - 1}$.
- Soit $i_1 \in 0, ..., N$ le plus grand indice tel que $g(-y_{max} + i_1 \cdot h) < y_{max}$. Entre $g(y_{max} + i_1 \cdot h)$ et y_{max}, l'espace est discrétisé de manière uniforme à l'aide de $N - i_1 + 1$ points $y_{i_1+1}, ..., y_N)$.
- Pour tout $i \in i_0, ..., i_1$, on définit $y_i = g(-y_{max} + ih)$.

On définit ensuite, pour $i \in 0, ..., N$:

$$\begin{aligned} f_i &= f(y_i), \\ h_i &= y_{i+1} - y_i. \end{aligned}$$

Soit $u(T) = (U(y_i, T))_{0 \le i \le N} \in I\!R^N$. Pour discrétiser $\tilde{A}$ par un schéma centré, on remplace

$$\begin{aligned} \frac{\partial U}{\partial y}(y_i, T) \quad &\text{par} \quad \frac{1}{2}\left(\frac{u_{i+1}(T) - u_i(T)}{h_i} + \frac{u_i(T) - u_{i-1}(T)}{h_{i-1}}\right), \\ \frac{\partial^2 U}{\partial y^2}(y_i, T) \quad &\text{par} \quad \frac{2}{h_i + h_{i-1}}\left(\frac{u_{i+1}(T) - u_i(T)}{h_i} - \frac{u_i(T) - u_{i-1}(T)}{h_{i-1}}\right). \end{aligned}$$

Posons, pour $i \in \{1, ..., N-1\}$:

$$\begin{aligned} \alpha_{i,T} &= -\frac{\hat{\sigma}^2(y_i, T)}{(h_i + h_{i-1})h_{i-1}} - \frac{1}{2h_{i-1}}\left(r - q + \frac{\hat{\sigma}^2(y_i, T)}{2}\right), \\ \beta_{i,T} &= \frac{\hat{\sigma}^2(y_i, T)}{h_i + h_{i-1}}\left(\frac{1}{h_i} + \frac{1}{h_{i-1}}\right) + \frac{1}{2}\left(r - q + \frac{\hat{\sigma}^2(y_i, T)}{2}\right)\left(\frac{1}{h_{i-1}} - \frac{1}{h_i}\right) + q, \\ \gamma_{i,T} &= -\frac{\hat{\sigma}^2(y_i, T)}{(h_i + h_{i-1})h_i} + \frac{1}{2h_i}\left(r - q + \frac{\hat{\sigma}^2(y_i, T)}{2}\right). \end{aligned}$$

On a alors, pour $i \in \{1, ..., N-1\}$ et pour tout T :

$$(\tilde{A}_T u(T))_i \quad = \quad \alpha_{i,T} u_{i-1}(T) + \beta_{i,T} u_i(T) + \gamma_{i,T} u_{i+1}(T).$$

Une fois ces changements effectués, la méthode de résolution de l'EDP est la même que dans le cas d'une discrétisation uniforme (cf. 2.4).

2.5.2 Tests numériques

On refait le même test que précédemment pour évaluer l'intérêt de cette nouvelle discrétisation (voir la figure 3). On note une nette diminution de l'erreur autour de S_0. Par contre, les résultats sont bien sûr moins bons aux extrémités, ce qui ne présente pas de problème dans la mesure où l'on s'intéresse aux prix des options dont le strike n'est pas trop éloigné du prix actuel de l'actif sous-jacent.

Dans l'exemple suivant, on choisit une volatilité qui dépend du temps :

$$\sigma(t) = \frac{t}{2}$$

et on augmente le nombre de pas d'espace (N=1000) et de temps (M=200). On obtient les résultats suivants montrés à la figure 4.

Cette méthode de pricing avec volatilité locale a également été validée par comparaison avec d'autres méthodes de pricing couramment utilisées en finance :

- méthode de Monte-Carlo,
- résolution de l'EDP de Black-Scholes généralisée avec pas adaptatif.

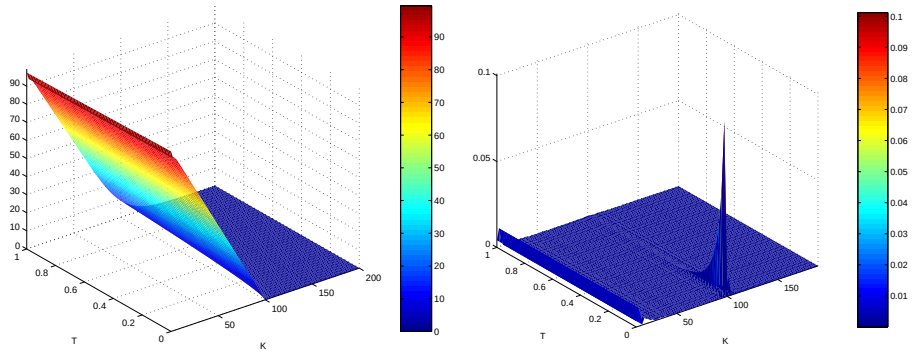


FIG. 3: $\sigma = 0.2$. *Discrétisation non uniforme en espace. A gauche : Nappe de prix obtenue par l'EDP de Dupire. A droite : Différences de prix entre Dupire et Black-Scholes.*

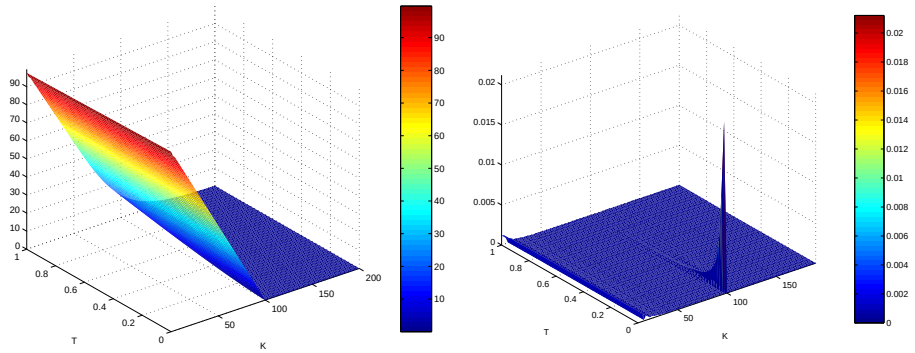


FIG. 4: $\sigma(t) = \frac{t}{2}$. *Discrétisation non uniforme en espace. A gauche : Nappe de prix obtenue par l'EDP de Dupire. A droite : Différences de prix entre Dupire et Black-Scholes.*

3 Problème inverse : description de l'algorithme de calibration

3.1 Formulation du problème

Nous avons décrit à la section 2 la résolution numérique de l'EDP de Dupire :

$$\frac{\partial U}{\partial T}(y,T) + (r - q + \frac{1}{2}\hat{\sigma}^2(y,T))\frac{\partial U}{\partial y}(y,T) - \frac{1}{2}\hat{\sigma}^2(y,T)\frac{\partial^2 U}{\partial y^2}(y,T) \quad = \quad -qU(y,T),$$

avec les conditions aux limites suivantes :

$$\begin{cases} U(y,t_0) = \max(S_0 - e^y, 0) & \text{pour tout réel } y, \\ \lim_{y\to-\infty} U(y,T) = S_0 e^{-q(T-t_0)} & \text{pour } t_0 \leq T, \\ \lim_{y\to+\infty} U(y,T) = 0 & \text{pour } t_0 \leq T. \end{cases}$$

Nous pouvons ainsi calculer, pour une volatilité donnée, le prix de l'option $V(K,T) = U(e^y,T)$ pour tout prix d'exercice K et toute maturité T.

A l'instant $t = t_0$, les données du marché correspondent à des prix d'options de maturités $T_1, ..., T_n$ et, pour chacune de ces maturités T_i, de prix d'exercice $K_{i1}, ..., K_{im_i}$. On note $\tilde{V}$ le vecteur formé de tous les prix d'options $\tilde{V}_{ij}$ de maturité T_i et de prix d'exercice K_{ij}.

Pour une volatilité σ donnée, on note $V(\sigma)$ le vecteur formé de tous les prix d'options $V_{ij}(\sigma)$ de maturité T_i et de prix d'exercice K_{ij}, calculés en résolvant l'EDP de Dupire. Notre objectif est de déterminer la volatilité σ telle que les prix calculés soient aussi proches que possible des données. De plus, nous souhaitons à priori (mais ce point reste à discuter) obtenir une nappe de volatilité aussi lisse que possible. Nous formulons le problème de calibration de la façon suivante :

$$\min_{\sigma} F(\sigma) : = \frac{1}{2}\|V(\sigma) - \tilde{V}\|^2 + \lambda \parallel \nabla\sigma \parallel^2 \tag{3}$$

où $\|.\|$ correspond à la norme Euclidienne dans l'espace discret des données et $\lambda \in I\!R$ est un coefficient de régularisation à déterminer. Le premier terme de la fonction coût F, noté

$$G(\sigma) = \frac{1}{2}\|V(\sigma) - \tilde{V}\|^2,$$

impose que les prix calculés collent aux données, alors que le second terme, noté

$$F_1(\sigma) = \parallel \nabla\sigma \parallel^2$$

impose que la volatilité ne soit pas trop irrégulière, ce qui permet de régulariser le problème inverse. Le choix de λ est un compromis entre précision et stabilité et il est en général difficile à déterminer en pratique. Nous verrons à la section 4 qu'il est possible de choisir $\lambda = 0$ et de régulariser d'une autre façon (voir l'approche multiéchelle décrite au paragraphe 4.2).

3.2 Paramétrage de la volatilité par des splines bicubiques

Avant de résoudre le problème d'optimisation (3), il faut paramétrer la volatilité σ, fonction de deux variables S et t, dans un espace de dimension finie adéquat. Si l'on garde la même grille de discrétisation que pour l'EDP, on aura trop de paramètres pour la variable d'optimisation. Mais rien ne nous oblige à prendre la même discrétisation pour U et pour σ. Nous choisissons de discrétiser σ sur une grille beaucoup plus grossière que la grille de différences finies à l'aide des splines bicubiques. Les deux principaux avantages sont les suivants :

- le nombre d'inconnues du problème d'optimisation, correspondant au nombre de degrés de liberté des splines bicubiques, est beaucoup plus petit ;
- la solution du problème d'optimisation sera plus régulière par construction car les splines bicubiques possèdent une régularité $\mathcal{C}^2$.

La grille grossière sur laquelle nous définissons les splines bicubiques correspond à une discrétisation du domaine $[y_{min}; y_{max}] \times [t_0; T_{max}]$ en n mailles suivant y et m mailles suivants T. Une grille $n \times m$ étant donnée, les splines bicubiques sont des polynômes de degré 3 par morceaux qui se raccordent de façon $\mathcal{C}^2$ entre chaque maille (voir De Boor [9]). Dans l'espace des splines bicubiques, les degrés de liberté correspondent aux :

- valeurs de la fonction σ en tous les nœuds de la grille ;
- valeurs des dérivées par rapport à y aux nœuds tels que $y = y_{min}$ et $y = y_{max}$;
- valeurs des dérivées par rapport à T aux nœuds tels que $T = t_0$ et $T = T_{max}$;
- valeurs des dérivées croisées aux quatre coins de la grille.

La figure 5 permet de visualiser les degrés de liberté sur une grille 3×3. Le nombre de degrés de liberté est égal à :

$$(n+1)(m+1) + 2(m+1) + 2(n+1) + 4 = (n+3)(m+3).$$

L'algorithme d'interpolation permettant de déterminer la valeur de la volatilité en n'importe quel point du domaine à partir des degrés de liberté est détaillé dans l'annexe A.

Nous décrirons à la section 4 comment choisir n et m, ce qui détermine le nombre d'inconnues du problème d'optimisation et la régularité plus ou moins importante donnée par construction à la volatilité.

3.3 Choix de l'algorithme d'optimisation de Quasi-Newton

Pour résoudre le problème d'optimisation 3, nous choisissons d'utiliser l'algorithme de Quasi-Newton (voir par exemple [4]) qui permet d'éviter de calculer le hessien de la fonction coût F. Cet algorithme, qui nécessite le calcul de la fonction et de son gradient $g(\sigma)$ en chaque itéré, peut se schématiser ainsi (la matrice W est une approximation du hessien) :

Etape 0 tolérance $= \epsilon$, $\sigma = \sigma_{init}$, $W = W_{init}$ definie positive.

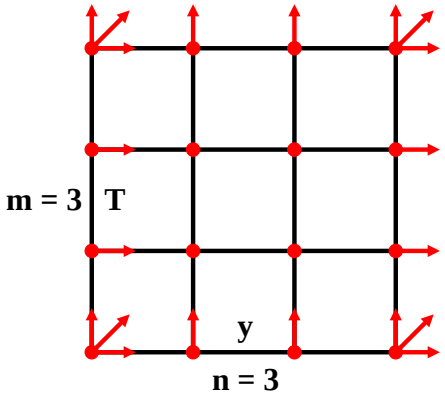


FIG. 5: *Degrés de liberté d'une spline bicubique sur une grille* 3×3.

Etape 1 Si $|g(\sigma)| \leq \epsilon$, STOP.

Etape 2 calcul d'une direction de descente d en résolvant le système linéaire : $Wd = -g(\sigma)$.

Etape 3 calcul du pas t le long de cette direction de descente, par recherche linéaire Wolfe.

Etape 4 mise à jour de l'itéré courant : $\sigma := \sigma + td$.

Etape 4 mise à jour de W par BFGS et retour à l'étape 1.

L'algorithme de Quasi-Newton utilisé dans la suite pour résoudre le problème (3) sans contraintes a été implémenté par [2]. Nous verrons à la section 4.3 qu'il peut être utile d'ajouter des contraintes de bornes sur la volatilité. Dans ce cas, nous utiliserons un algorithme de Quasi-Newton avec bornes développé par [6, 14].

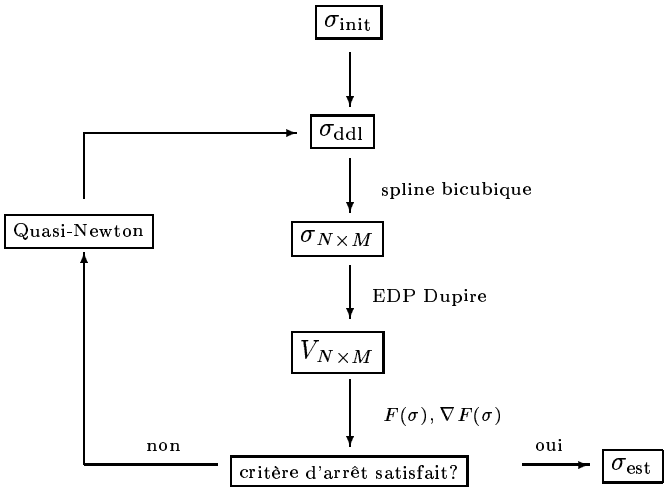
Dans chaque cas, il est nécessaire de savoir évaluer le gradient de la fonction coût

$$F(\sigma) = G(\sigma) + \lambda F_1(\sigma).$$

Il est préférable de calculer à chaque étape ce gradient de manière analytique plutôt que par différences finies : le gain de temps est considérable, surtout lorsque le nombre de variables augmente. Nous montrons dans les annexes C et B comment calculer analytiquement les gradients, respectivement de G et de F_1.

3.4 Résumé de l'algorithme de calibration

Nous choisissons donc d'utiliser la méthode de Quasi-Newton couplée avec une interpolation par spline bicubique pour résoudre notre problème. La fonction de volatilité σ sera donc représentée par les degrés de liberté d'une spline bicubique sur une grille grossière de taille $n \times m$. Le schéma suivant résume les différentes étapes de l'algorithme de calibration :



3.5 Comparaison avec d'autres méthodes de calibration

Notre algorithme de calibration présente plusieurs différences avec les méthodes de calibration similaires (avec une approche de type problème inverse) déjà développées par Lagnado-Osher [12], Jackson *et al.* [11] et Coleman *et al.* [8].

Pour la résolution du problème direct, nous avons choisi d'utiliser l'EDP de Dupire et non celle de Black-Scholes généralisée. Dans l'EDP de Black-Scholes, les paramètres du problème sont le strike K et la maturité T de l'option considérée, et les variables sont t et S_0. Dans l'EDP de Dupire, le point de vue est différent : on se place en un point (S_0, t_0) (paramètres de problème) et les variables sont K et T. Par conséquent, lorsque l'on résout l'EDP de Dupire, on obtient une nappe de prix d'options sur K et T pour (S_0, t_0) fixés. Une seule résolution de cette EDP suffit donc à évaluer la fonction coût $F(\sigma)$, alors qu'il faudrait autant de résolutions de l'EDP de Black-Scholes qu'il y a de données du marché. Le gain de temps paraît non négligeable.

Concernant le terme de régularisation, nous utilisons le même terme que [12], à savoir $\lambda \parallel \nabla\sigma \parallel^2$, alors que [11] n'utilise aucun terme et [8] régularise le problème en ajoutant à la fonction coût un terme $r(\sigma)$ qui correspond à la valeur moyenne du gradient.

Nous avons choisi de représenter σ par les degrés de liberté d'une spline bicubique : de cette manière, beaucoup de fonctions lisses peuvent être représentées. Par comparaison, les représentations par splines proposées par [11] et [8] permettent d'obtenir un moins grand nombre de fonctions. En effet, dans [8], il s'agit de splines bicubiques avec moins de degrés de liberté car on impose que la dérivée seconde est nulle sur les bords du domaine ("natural splines") ; dans [11], il s'agit de splines cubiques par rapport à S et de fonctions linéaires par morceaux par rapport à t. La représentation choisie par [12] (valeurs de σ aux points de la grille de discrétisation de l'EDP) implique un problème de grande taille surdéterminé.

Enfin, nous utilisons un algorithme d'optimisation de Quasi-Newton avec ou sans bornes. De même, [11] utilise un algorithme de Quasi-Newton avec bornes, alors que [12] utilise une méthode de gradient sans bornes et [8] a choisi une méthode de points intérieurs avec région de confiance pour résoudre le problème avec bornes.

4 Résultats numériques de calibration

4.1 Tests sur données synthétiques : choix de la grille $n \times m$ et du coefficient de régularisation λ

Dans un premier temps, nous testons l'algorithme de calibration sur données synthétiques, c'est-à-dire sur des données générées avec notre modèle en résolvant l'équation de Dupire. Nous appelons "σ_{true}" la "vraie" volatilité que nous utilisons pour créer les données. Ces données correspondent à des call européens avec :

- $S_0 = 100$;
- $r = 0.05$;
- $q = 0.02$.

Concernant la résolution numérique de l'EDP de Dupire, nous choisissons les paramètres suivants :

- $[y_{min}; y_{max}] = [-5.298; 5.298]$ (donc $[K_{min}; K_{max}] = [0.005; 200]$) est discrétisé par $N = 400$ points (grille régulière) ;
- $[t_0; T_{max}] = [0; 1]$ est discrétisé par $M = 100$ points ;
- $\theta = 0.5$.

Nous avons généré 22 données avec la vraie volatilité suivante :

$$\sigma_{true}(S, t) = \frac{15}{S}$$

pour 11 prix d'exercice et 2 maturités :

- $K = [90, 92, 94, ..., 110]$;
- $T = [0.5, 1]$.

En ce qui concerne l'optimisation, nous effectuons 30 itérations de l'algorithme de Quasi-Newton sans contraintes, en partant d'une volatilité initiale constante :

$$\sigma_{init} = 0.15.$$

Pour chaque test d'inversion, nous comparons la vraie volatilité à la volatilité estimée avec notre algorithme de calibration. Nous montrons également l'erreur relative entre les deux volatilités ainsi que la décroissance du critère au cours des itérations.

Les premiers tests (figures 6, 7 et 8) correspondent à une grille de paramétrage de σ très grossière avec :

$$n = m = 1$$

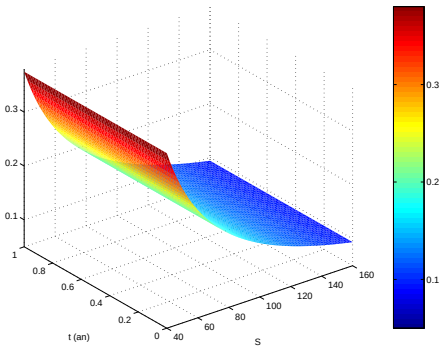
(donc 16 degrés de liberté) et à différentes valeurs de λ : 0, 10 et 100. On constate que le meilleur résultat est obtenu avec $\lambda = 0$. Le fait d'ajouter un terme de régularisation n'a rien amélioré sur cet exemple.

Avec $\lambda = 0$, nous choisissons de discrétiser σ sur une grille plus grossière avec :

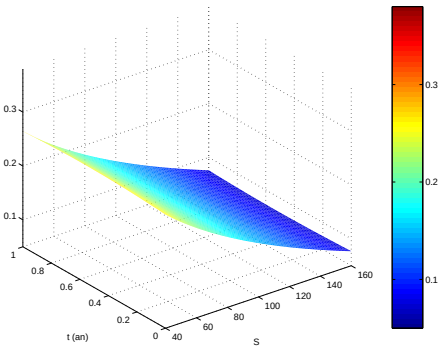
$$n = m = 3$$

(36 degrés de liberté). Le résultat montré à la figure 9 est moins bon que celui obtenu avec une grille 1×1 ; c'est très net lorsque l'on observe l'erreur relative.

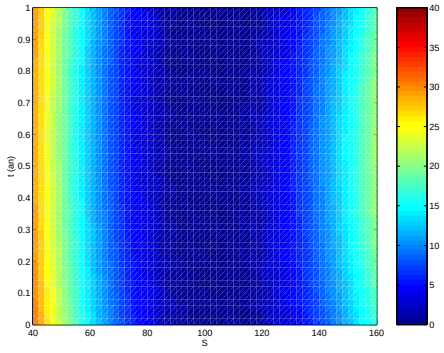
Compte tenu de ces premiers résultats, nous choisirons dans la suite une grille très grossière ($n = m = 1$) et un coefficient de régularisation $\lambda = 0$.



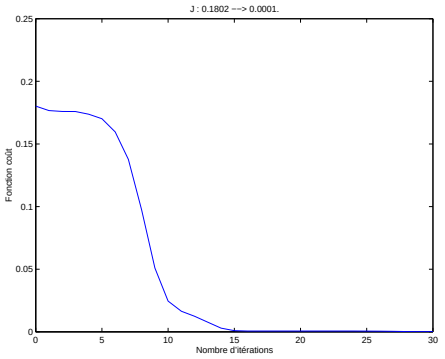
(a) Vraie volatilité.



(b) Volatilité estimée.

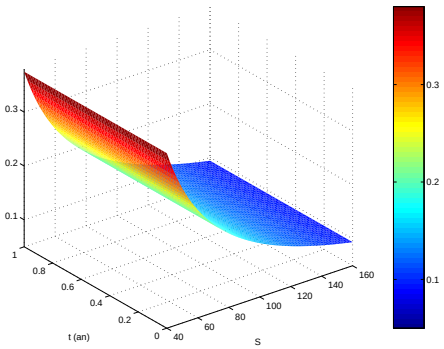


(c) Erreur relative (en %).

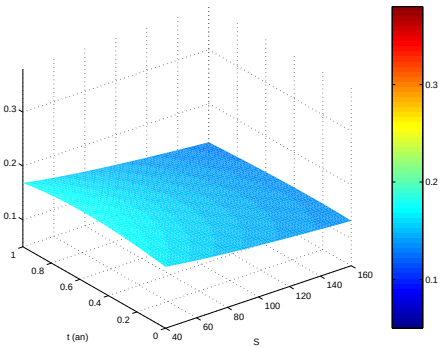


(d) Décroissance du critère.

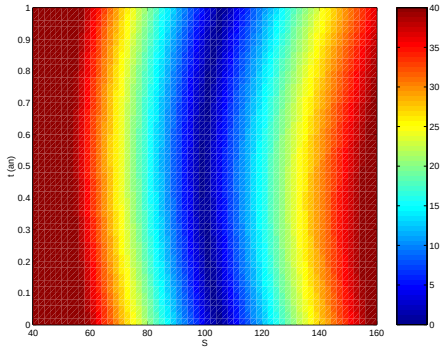
FIG. 6: *Calibration avec* $n = m = 1$ *et* $\lambda = 0$.



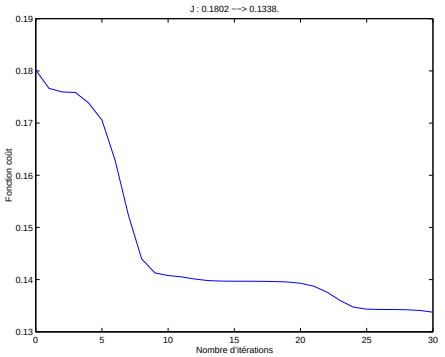
(a) Vraie volatilité.



(b) Volatilité estimée.

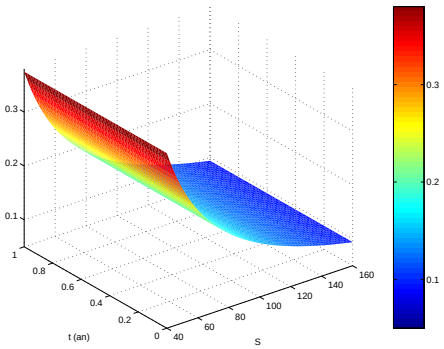


(c) Erreur relative (en %).

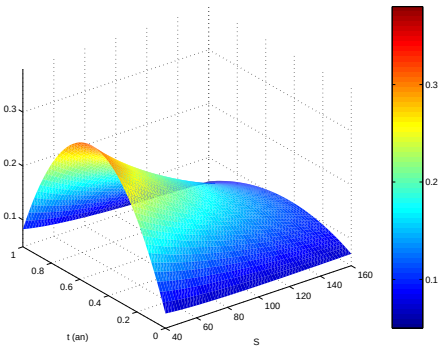


(d) Décroissance du critère.

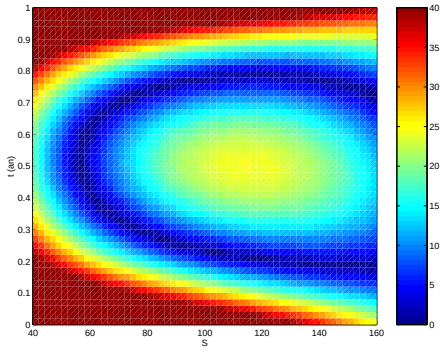
FIG. 7: *Calibration avec* $n = m = 1$ *et* $\lambda = 10$.



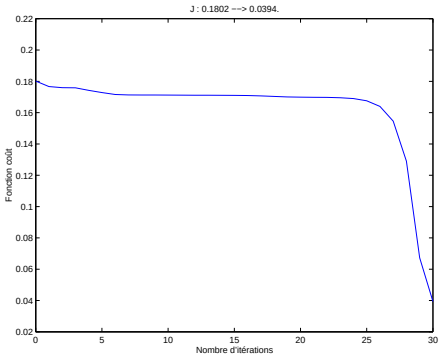
(a) Vraie volatilité.



(b) Volatilité estimée.

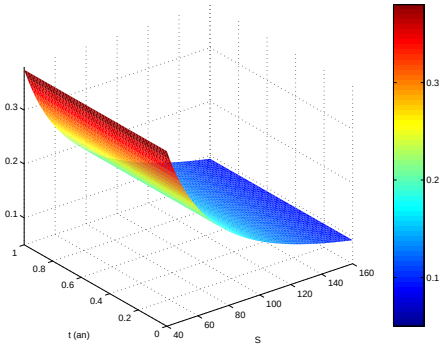


(c) Erreur relative (en %).

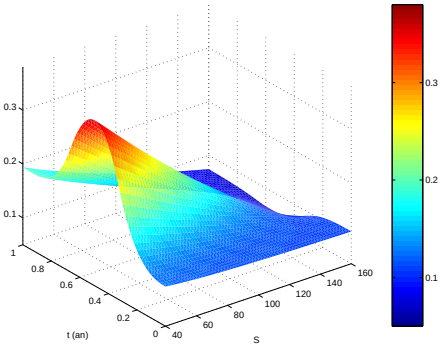


(d) Décroissance du critère.

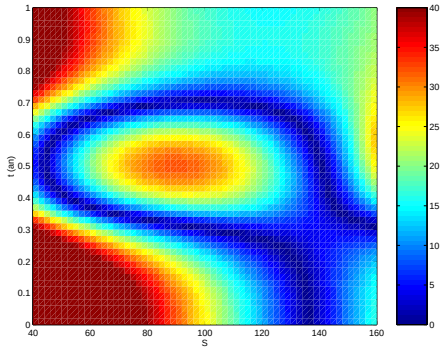
FIG. 8: *Calibration avec* $n = m = 1$ *et* $\lambda = 100$.



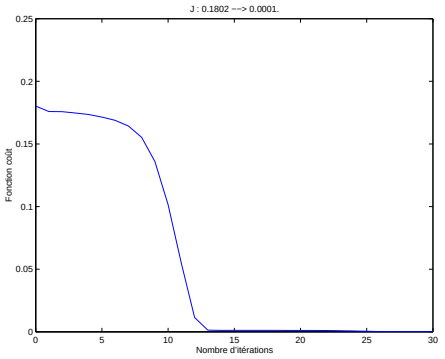
(a) Vraie volatilité.



(b) Volatilité estimée.



(c) Erreur relative (en %).



(d) Décroissance du critère.

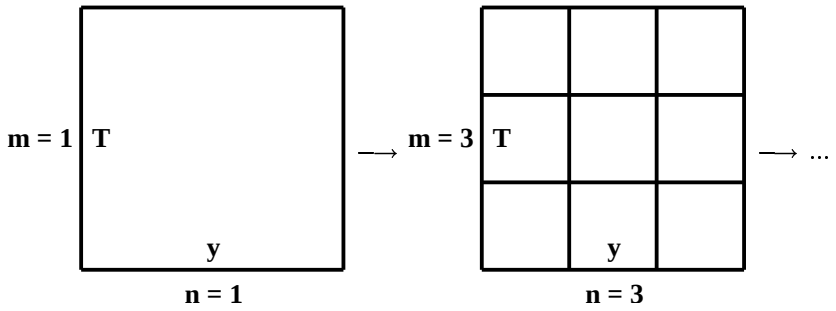
FIG. 9: *Calibration avec* $n = m = 3$ *et* $\lambda = 0$.

4.2 Régularisation par une approche multiéchelle

Avec le choix d'une grille de paramétrage de σ très grossière ($n = m = 1$), nous retrouvons une volatilité très régulière par construction puisque l'on ne peut pas représenter des volatilités très irrégulières. La contre-partie est qu'il est difficile d'affiner σ dans certaines régions du domaine, ce qui permettrait de mieux coller aux données.

Pour résoudre ce problème, nous choisissons d'utiliser une approche multiéchelle. Cette approche a déjà utilisée sur des problèmes inverses issus de la Géophysique ; voir par exemple Chavent *et al.* [5] ou Clément [7]. Le principe de la méthode est de rechercher dans un premier temps l'allure générale de la volatilité sur une grille très grossière, puis d'optimiser sur une grille plus fine en partant du point initial que l'on vient d'obtenir, et ainsi de suite. Par exemple :

$$\sigma_{init}^{1\times 1} \longrightarrow (\text{Q.} - \text{Newton}) \longrightarrow \sigma_{est}^{1\times 1} \rightarrow \sigma_{init}^{3\times 3} \longrightarrow (\text{Q.} - \text{Newton}) \longrightarrow \sigma_{est}^{3\times 3} \rightarrow ...$$



Nous allons appliquer cette méthode sur les prochains résultats sur données synthétiques et sur données réelles.

4.3 Tests sur données synthétiques : influence du point initial, choix d'un algorithme de Quasi-Newton avec bornes

Nous avons généré de nouvelles données synthétiques correspondant à des call européens avec :

- $S_0 = 100$;
- $r = 0.05$;
- $q = 0.02$,

avec le même choix de grille de différences finies que dans l'exemple précédent :

- $[y_{min}; y_{max}] = [-5.298; 5.298]$ (donc $[K_{min}; K_{max}] = [0.005; 200]$) est discrétisé par $N = 400$ points (grille régulière) ;
- $[t_0; T_{max}] = [0; 1]$ est discrétisé par $M = 100$ points ;

– $\theta = 0.5$.

La vraie volatilité utilisée est :

$$\sigma_{true}(S,t) = 0.05 + 0.1 \exp\left(-\frac{S}{100}\right) + 0.5t$$

et nous avons simulé 22 données correspondant aux couples (K,T) suivants :

– $K = [90, 92, 94, ..., 110]$;

– $T = [0.5, 1]$.

Nous avons utilisé l'approche multiéchelle décrite au paragraphe précédent, en passant d'une grille 1×1 (30 itérations) à une grille 3×3 (30 itérations).

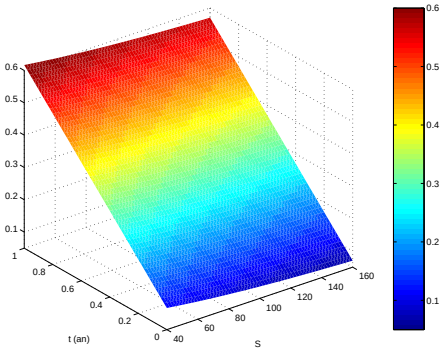
Nous avons testé la robustesse de l'algorithme en partant de différentes volatilités initiales (constantes) :

$$\sigma_{init} = 0.1,\ 0.35,\ ,0.5 \text{ et } 0.9.$$

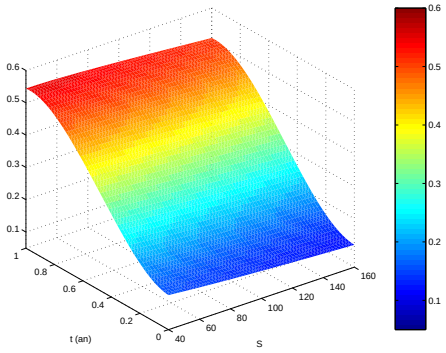
Les résultats obtenus après utilisation de l'approche multiéchelle (sur la grille 3×3) sont montrés respectivement aux figures 10, 11, 12 et 13. Notons que dans chaque cas, la volatilité estimée sur la grille 3×3 (après multiéchelle) est proche (graphiquement) de celle estimée sur la grille 1×1, c'est pourquoi nous ne montrons pas cette dernière. En revanche, l'approche multiéchelle a permis de faire décroître encore la fonction coût et ainsi de mieux reproduire les données. Les trois premiers résultats obtenus sont bons puisque l'erreur relative entre la vraie volatilité et la volatilité estimée est inférieure à 5% sur une grande partie du domaine. Si on part du point initial très éloigné $\sigma_{init} = 0.9$ (figure 13), on constate que la volatilité estimée possède des valeurs négatives pour des maturités inférieures à 0.5. Notons que ce n'est pas très gênant puisque c'est σ^2 qui intervient dans les calculs et non σ. Cependant, il est possible de résoudre ce problème en ajoutant des contraintes de bornes sur σ dans le problème d'optimisation.

Par conséquent, nous avons choisi de remplacer l'algorithme de Quasi-Newton sans contraintes utilisé jusqu'à présent par un algorithme de Quasi-Newton (à mémoire limitée) avec contraintes de bornes. Il s'agit d'un algorithme développé par Nocedal *et al.* (voir [6, 14]). Notons que les contraintes de bornes sont imposées sur les degrés de liberté de σ correspondant aux valeurs de la fonction en chaque nœud. Nous pouvons remarquer que ceci ne garantit pas que la fonction σ n'ait pas de valeurs négatives à l'intérieur du domaine puisque les degrés de liberté correpondant aux dérivées sont laissés libres.

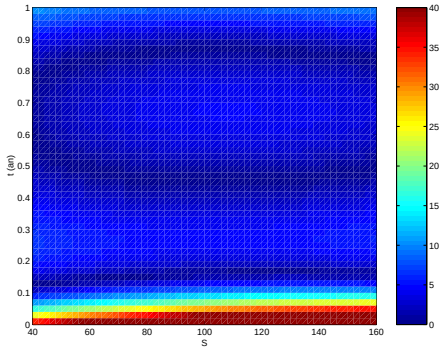
En choisissant les bornes min et max à 0 et à 1, nous obtenons le résultat montré à la figure 14. La volatilité estimée ne contient plus de valeurs négatives et l'erreur est inférieure à 5% sur une grande partie du domaine. Nous avons également refait les tests de calibration en partant de 0.1, 0.35 et 0.5 avec l'algorithme de Quasi-Newton avec bornes et les résultats sont similaires à ceux obtenus précédemment. Ainsi, en choisissant l'algorithme avec bornes, nous obtenons des volatilités estimées très satisfaisantes quel que soit le point initial choisi (0.1, 0.35, 0.5 ou 0.9).



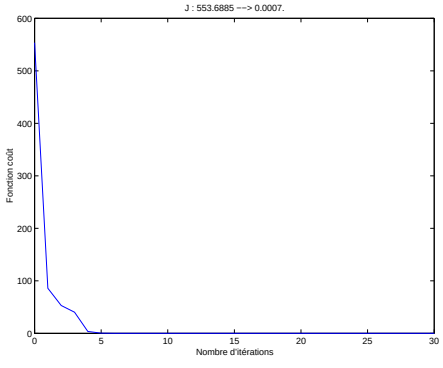
(a) Vraie volatilité.



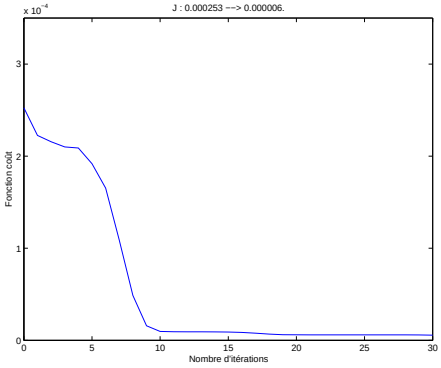
(b) Volatilité estimée.



(c) Erreur relative (en %).

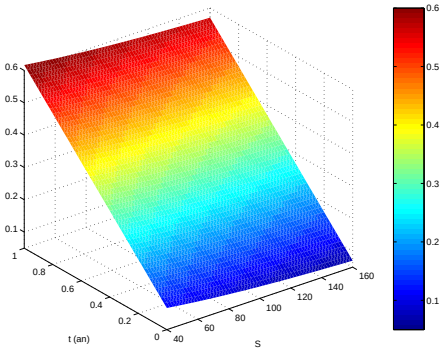


(d) Décroissance du critère : 1×1.

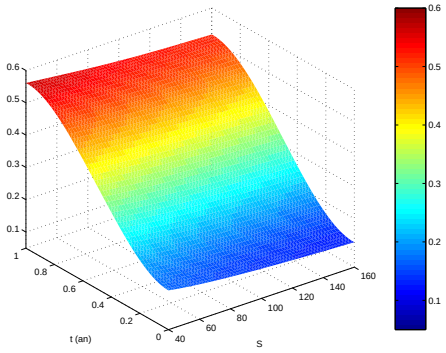


(e) Décroissance du critère : 3×3.

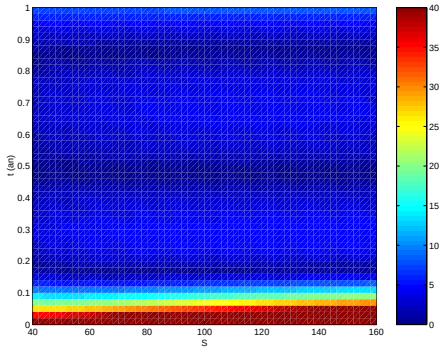
FIG. 10: *Calibration en partant de* $\sigma_{init} = 0.1$.



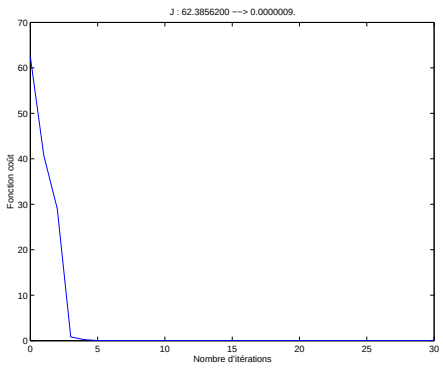
(a) Vraie volatilité.



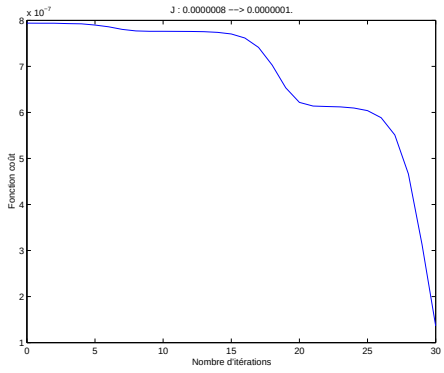
(b) Volatilité estimée.



(c) Erreur relative (en %).

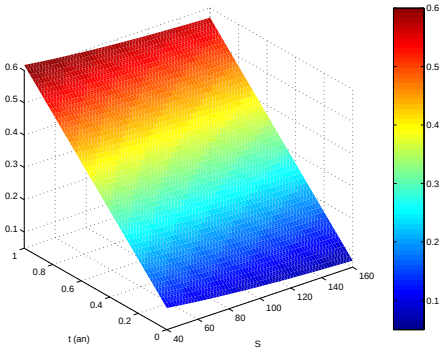


(d) Décroissance du critère : 1×1.

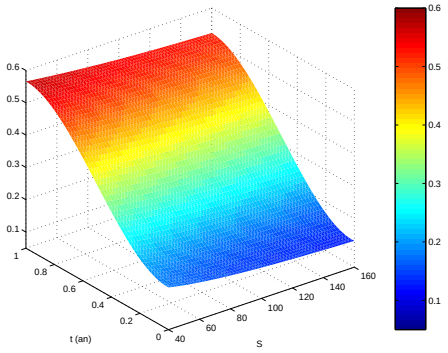


(e) Décroissance du critère : 3×3.

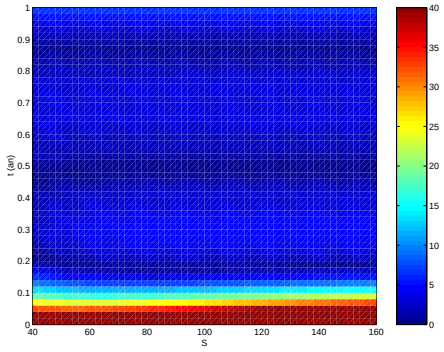
FIG. 11: *Calibration en partant de* $\sigma_{init} = 0.35$.



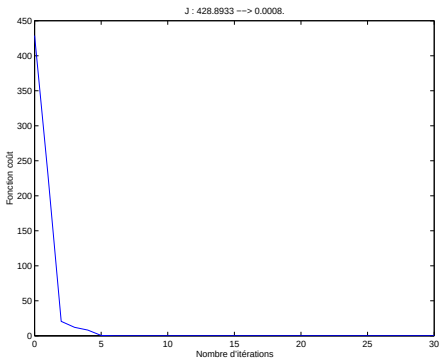
(a) Vraie volatilité.



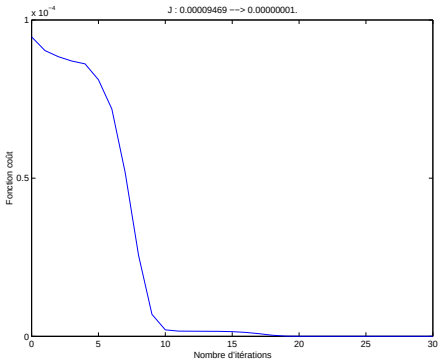
(b) Volatilité estimée.



(c) Erreur relative (en %).

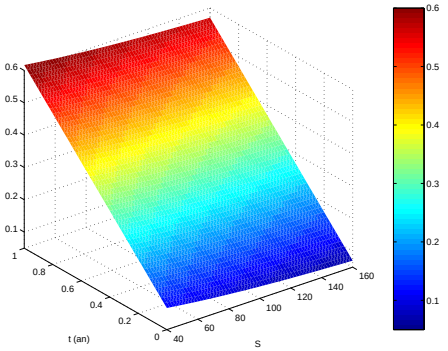


(d) Décroissance du critère : 1×1.

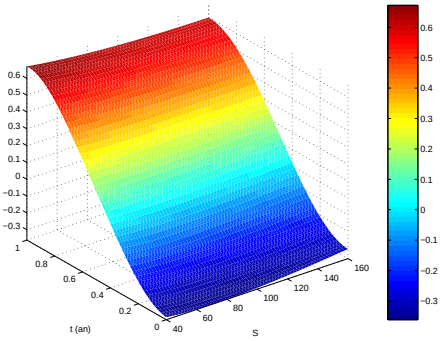


(e) Décroissance du critère : 3×3.

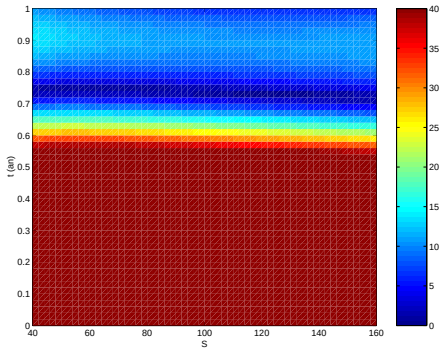
FIG. 12: *Calibration en partant de* $\sigma_{init} = 0.5$.



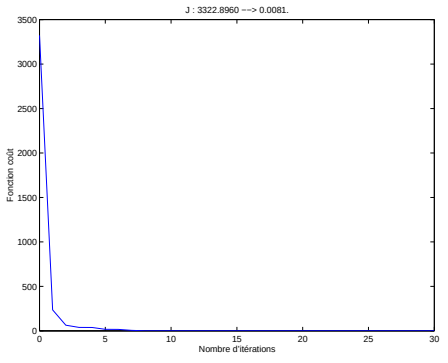
(a) Vraie volatilité.



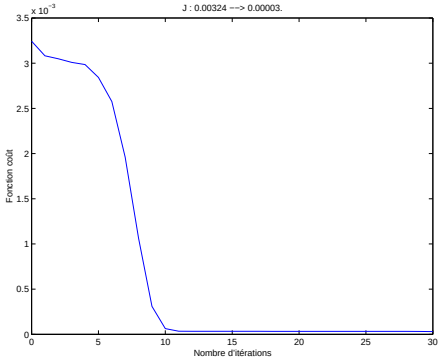
(b) Volatilité estimée.



(c) Erreur relative (en %).

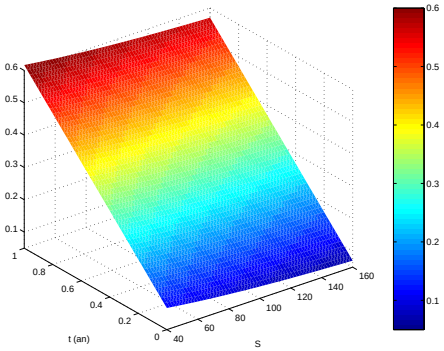


(d) Décroissance du critère : 1×1.

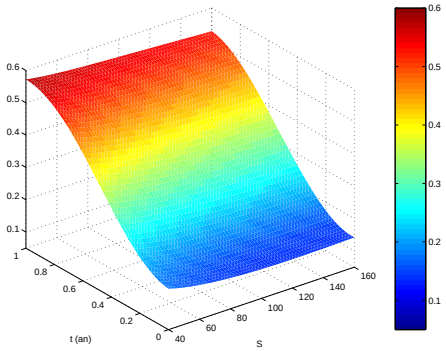


(e) Décroissance du critère : 3×3.

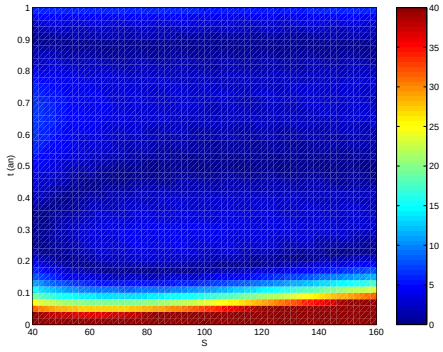
FIG. 13: *Calibration en partant de* $\sigma_{init} = 0.9$.



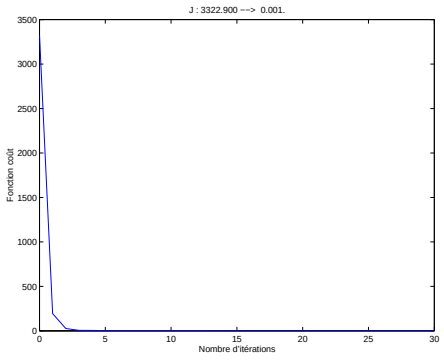
(a) Vraie volatilité.



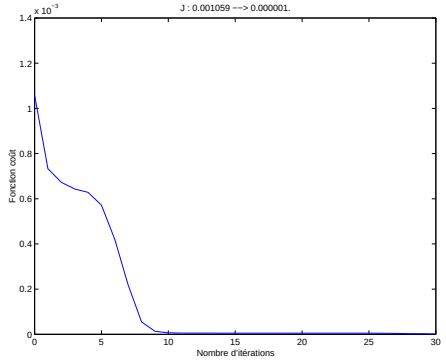
(b) Volatilité estimée.



(c) Erreur relative (en %).



(d) Décroissance du critère : 1×1.



(e) Décroissance du critère : 3×3.

FIG. 14: *Calibration en partant de $\sigma_{init} = 0.9$ avec un algorithme de Quasi-Newton avec bornes (0-1).*

4.4 Tests sur données réelles du marché

Nous avons ensuite testé notre algorithme de calibration sur des données réelles récupérées dans l'article de Coleman *et al.* [8]. Il s'agit de 70 call européens provenant du "European S&P 500 index" avec :

- $S_0 = 590$;
- $r = 0.06$;
- $q = 0.0262$;
- 10 prix d'exercice : $K = [501.5, 531, 560.5, 590, 619.5, 649, 678.5, 708, 767, 826]$;
- 7 maturités : $T = [0.175, 0.425, 0.695, 0.94, 1, 1.5, 2]$.

Nous avons choisi $\lambda = 0$ et le point initial suivant :

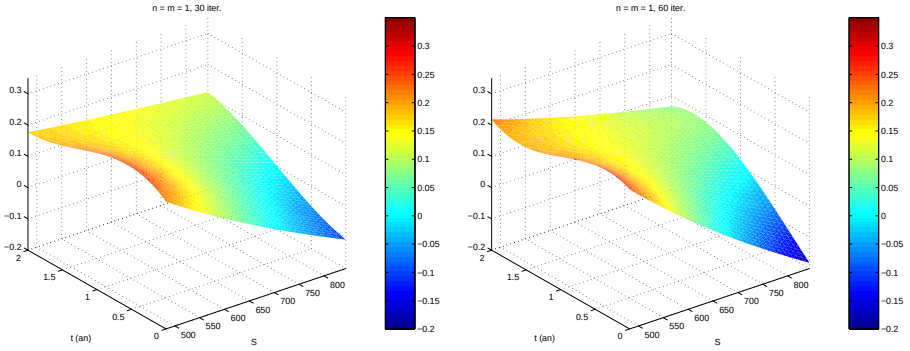
$$\sigma_{init} = 0.13$$

correspondant à la moyenne des volatilités implicites. Nous comparons les résultats obtenus :

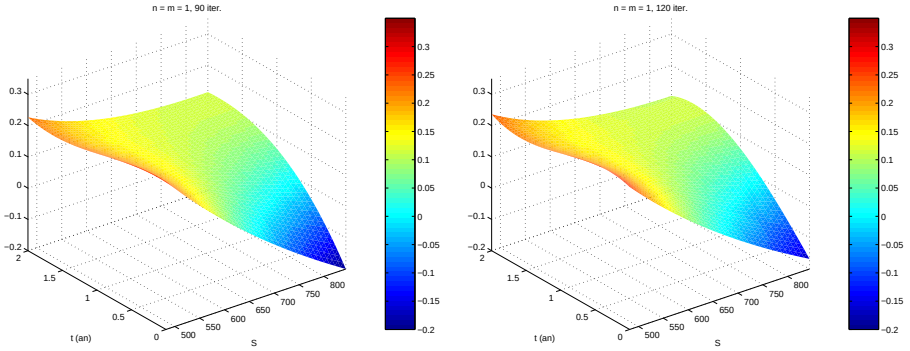
- sans l'approche multiéchelle : 120 itérations sur une grille 1×1 ;
- avec l'approche multiéchelle : 4×30 itérations sur des grilles 1×1, 3×3, 6×6 puis 12×12.

Nous avons utilisé dans un premier temps l'algorithme de Quasi-Newton sans contraintes. Les résultats obtenus sans et avec l'approche multiéchelle sont montrés respectivement aux figures 15 et 16. On constate qu'avec l'approche multiéchelle, la fonction coût descend plus bas (cf. figure 19 à gauche), donc on reproduit mieux les données avec cette volatilité estimée. On constate qu'entre la grille 1×1 et la grille 12×12, l'allure générale est conservée et la volatilité estimée sur la grille 12×12 comporte de légères variations permettant de mieux expliquer les données dans certaines zones. Cependant, il y a des valeurs négatives dans une petite partie du domaine où la volatilité influe peu sur les données.

Pour résoudre ce problème, nous utilisons à nouveau l'algorithme de Quasi-Newton avec bornes. En fixant les bornes min et max à 0 et à 1, nous obtenons encore des valeurs négatives dans certaines parties du domaine. Cependant, en augmentant la borne min à 0.08, la volatilité estimée ne contient plus de valeurs négatives, comme le montrent les résultats obtenus sans et avec l'approche multiéchelle (figures 17 et 18). A nouveau, l'approche multiéchelle permet de faire décroître la fonction coût vers une valeur plus basse (cf. figure 19 à droite), proche de la valeur obtenue avec l'algorithme de Quasi-Newton sans contraintes (1.69 au lieu de 1.58). La volatilité estimée après utilisation de l'approche multiéchelle (sur la grille 12×12) possède des valeurs comprises entre 0.05 et 0.3 qui ne semblent pas aberrantes. Comparée aux résultats obtenus par Coleman *et al.* [8], la volatilité que nous avons estimée possède des oscillations comparables entre 0.1 et 0.25, mais elle ne possède pas de grandes valeurs (proches de 0.7) pour des maturités proches de 0.

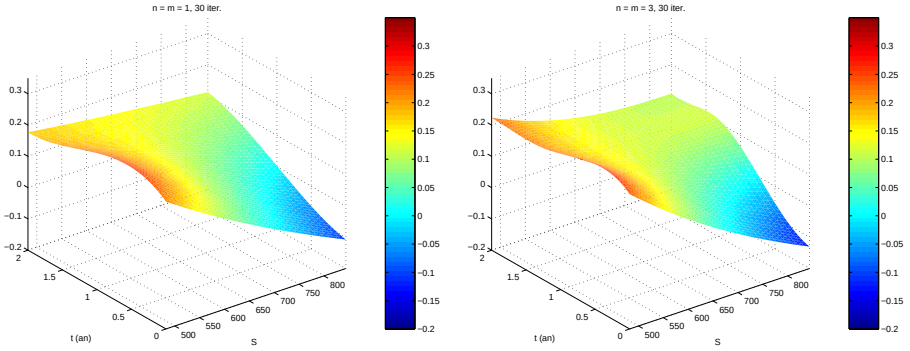


(a) 30 itérations. (b) 60 itérations.



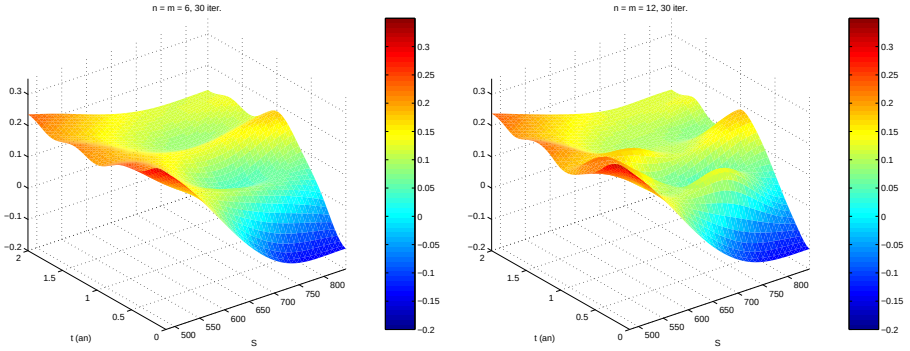
(c) 90 itérations. (d) 120 itérations.

FIG. 15: *Calibration sur données réelles avec* $n = m = 1$ *sans utiliser l'approche multiéchelle. Algorithme de Quasi-Newton sans contraintes.*
F : 263.5 $\searrow$ 5.36 $\searrow$ 4.63 $\searrow$ 3.27 $\searrow$ 3.24.



(a) 30 itérations : grille 1×1.

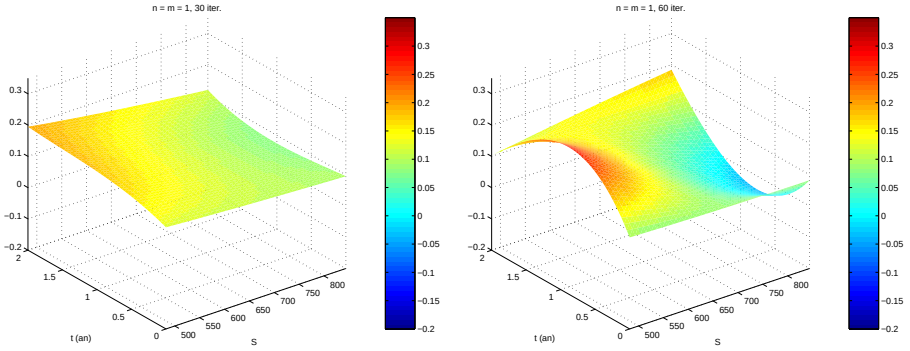
(b) 30 itérations : grille 3×3.



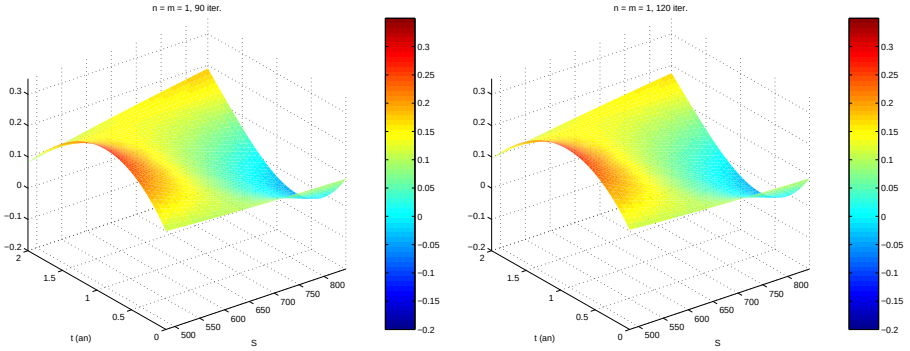
(c) 30 itérations : grille 6×6.

(d) 30 itérations : grille 12×12.

FIG. 16: *Calibration sur données réelles en utilisant l'approche multiéchelle. Algorithme de Quasi-Newton sans contraintes.*
F : 263.5 $\searrow$ 5.36 $\searrow$ 3.20 $\searrow$ 2.09 $\searrow$ 1.58.

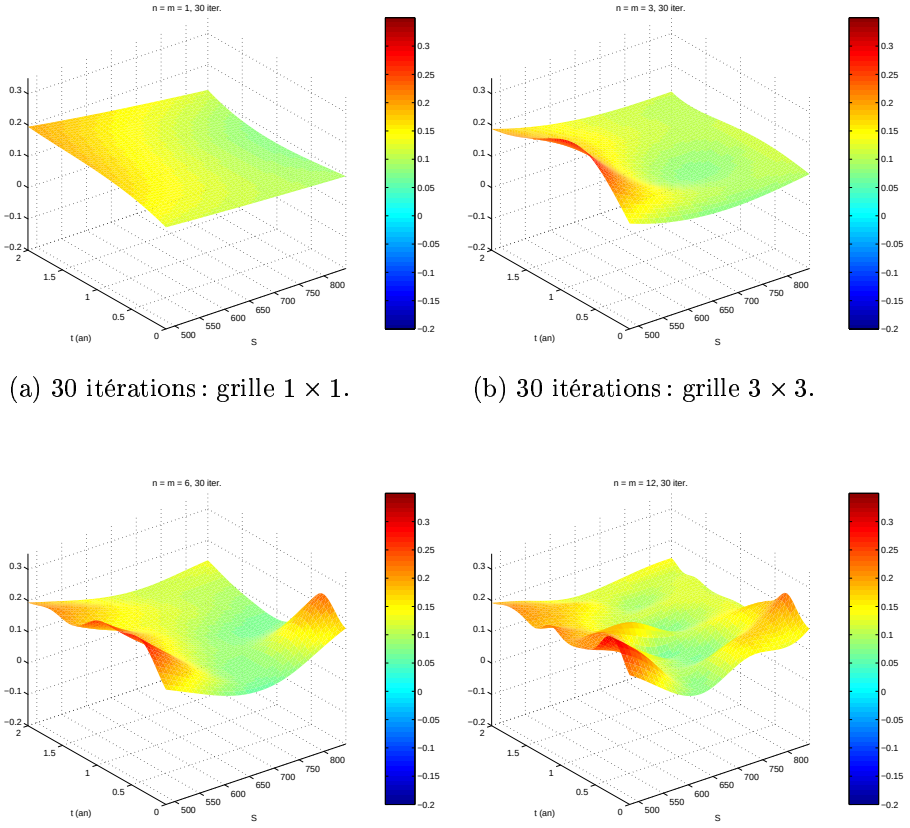


(a) 30 itérations. (b) 60 itérations.



(c) 90 itérations. (d) 120 itérations.

FIG. 17: *Calibration sur données réelles avec* $n = m = 1$ *sans utiliser l'approche multiéchelle. Algorithme de Quasi-Newton avec bornes (0.08-1).*
F : 263.5 $\searrow$ 90.47 $\searrow$ 13.21 $\searrow$ 11.30 $\searrow$ 10.95.



(a) 30 itérations : grille 1×1.

(b) 30 itérations : grille 3×3.

(c) 30 itérations : grille 6×6.

(d) 30 itérations : grille 12×12.

FIG. 18: *Calibration sur données réelles en utilisant l'approche multiéchelle. Algorithme de Quasi-Newton avec bornes (0.08-1).*
F : 263.5 $\searrow$ 90.47 $\searrow$ 10.06 $\searrow$ 3.66 $\searrow$ 1.69.

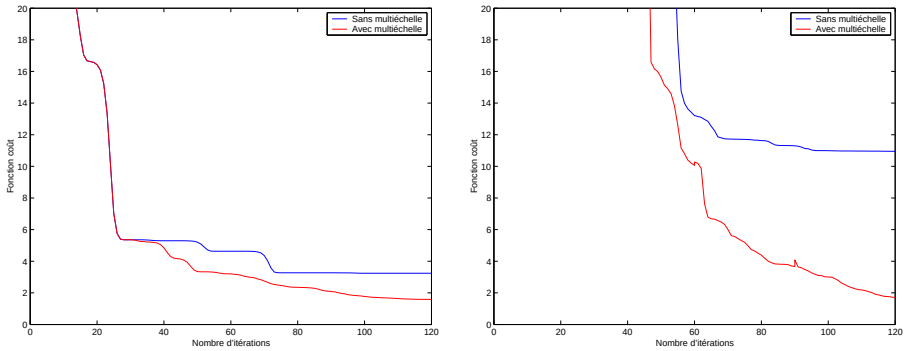


FIG. 19: *Décroissance du critère avec ou sans l'approche multiéchelle. A gauche : Quasi-Newton sans contraintes. A droite : Quasi-Newton avec bornes.*

5 Conclusion

La méthode de calibration proposée se ramène à la résolution d'un problème inverse régularisé. Elle se distingue des articles dont elle s'inspire ([12], [11], [8]) sur plusieurs points. Le problème direct de pricing repose sur l'EDP de Dupire et non sur celle de Black-Scholes, ce qui permet d'évaluer la fonction coût en une seule résolution d'EDP, alors qu'il faut résoudre autant d'EDP de Black-Scholes qu'il y a de données pour faire cette même évaluation. Une autre particularité réside dans la représentation de σ : nous utilisons ici des splines bicubiques, qui permettent de définir des fonctions C^2 très diverses. Enfin, nous avons choisi de calculer le gradient de la fonction coût analytiquement afin d'accélérer l'algorithme.

L'exploitation des résultats numériques nous a amené à faire une autre modification : nous choisissons un coefficient de régularisation $\lambda = 0$ et nous régularisons le problème en utilisant une approche multiéchelle. L'idée est d'obtenir une première volatilité très régulière sur une grille très grossière, puis de raffiner progressivement la grille de représentation de σ pour affiner la solution.

Les tests numériques effectués sur données synthétiques ont également montrés qu'il était important d'utiliser un algorithme de Quasi-Newton avec bornes pour éviter d'obtenir des valeurs négatives de σ dans certaines zones du domaine. Nous montrons que notre algorithme de calibration converge vers une solution très satisfaisante en partant de points initiaux (constants) très éloignés : 0.1, 0.35, 0.5 et 0.9. Sur les données réelles, nous montrons l'intérêt d'utiliser l'approche multiéchelle et l'algorithme de Quasi-Newton avec bornes. La volatilité estimée est assez régulière et les valeurs de σ ne semblent pas aberrantes.

Remerciements

Les auteurs remercient Agnès Sulem et Bernard Lapeyre pour les remarques constructives et l'intérêt qu'ils ont porté à ce travail.

A Interpolation par splines bicubiques

Nous montrons dans cette annexe comment déterminer la valeur d'une spline bicubique en un point quelconque du domaine à partir de ses degrés de liberté. Le détail des démonstrations se trouve dans l'article [9]. On considère une grille rectangulaire de points (x_i, y_j), $i = 0, ..., I$, $j = 0, ..., J$. Soient :

- $u_{ij} = u(x_i, y_j)$ donnés pour $i = 0, ..., I$ et $j = 0, ..., J$,
- $p_{ij} = u_x(x_i, y_j)$ donnés pour $i = 0, I$ et $j = 0, ..., J$,
- $q_{ij} = u_y(x_i, y_j)$ donnés pour $i = 0, ..., I$ et $j = 0, J$,
- $s_{ij} = u_{xy}(x_i, y_j)$ donnés aux 4 coins de la grille, i.e. pour $i = 0, I$ et $j = 0, J$.

Le problème consiste à interpoler ces points à l'aide d'une fonction $u \in C^2$.

A.1 Splines cubiques en dimension 1

Pour les fonctions d'une variable, l'interpolation par spline définit une fonction $u(x)$ de classe C^2 étant données :

- les valeurs $u(x_i)$ aux points x_i, $i = 0, ..., I$, $x_0 < ... < x_I$,
- les dérivées $p_i = u'(x_i)$ en x_0 et x_I.

La fonction d'interpolation est une fonction polynomiale cubique sur chacun des intervalles $[x_{i-1}, x_i]$, $i = 1, ..., I$.

Soient $\Delta x_i = x_{i+1} - x_i$ et $\Delta u_i = u_{i+1} - u_i$.

Théorème 1 *Soit u une fonction polynomiale cubique par morceaux de classe C^1. Pour des valeurs $u(x_i)$, $i = 1, ..., I$ et $u'(x_0), u'(x_I)$ données, il existe un unique ensemble de valeurs $p_i = u'(x_i)$, $i = 1, ..., I-1$ tel que $u \in C^2$. Ces valeurs sont déterminées par le système d'équations suivant :*

$$\Delta x_i p_{i-1} + 2(\Delta x_i + \Delta x_{i-1}) p_i + \Delta x_{i-1} p_{i+1} = 3\left[\frac{\Delta x_{i-1}}{\Delta x_i}\Delta u_i + \frac{\Delta x_i}{\Delta x_{i-1}}\Delta u_{i-1}\right], \ i = 1, ..., I-1. \tag{4}$$

Par conséquent, pour chaque ensemble de données $\{u_0, ..., u_I, p_0, p_I\}$, il existe une unique fonction u polynomiale cubique par morceaux de classe C^1 telle que :

$$u(x_i) = u_i, \qquad i = 0, ..., I, \qquad u'(x_0) = p_0, \qquad u'(x_I) = p_I.$$

Corrolaire 1 *$S(x; x_0, ..., x_I)$ est un espace de dimension $(I+3)$.*

Corrolaire 2 *L'ensemble des fonctions* $\Phi_i(x) \in S(x; x_0, ..., x_I)$, $i = 0, ..., I+2$ *définies par les conditions :*

$$\begin{array}{l}\Phi_j(x_i) = \delta_{ij} \text{ pour } i, j = 0, ..., I, \\ \Phi'_j(x_0) = \Phi'_j(x_I) = 0 \text{ pour } j = 0, ..., I, \\ \Phi_{I+1}(x_i) = \Phi_{I+2}(x_i) = 0 \text{ pour } i = 0, ..., I, \\ \Phi'_{I+1}(x_0) = \Phi'_{I+2}(x_I) = 1, \\ \Phi'_{I+1}(x_I) = \Phi'_{I+2}(x_0) = 0\end{array}$$

est une base de $S(x; x_0, ..., x_I)$.

A.2 Splines bicubiques en dimension 2

Soit $\{\psi_j(y)\}_{0 \le j \le J+2}$ la base de $S(y; y_0, ..., y_I)$ définie dans le corrolaire 2 précédent. Considérons le produit tensoriel $T = S(x; x_0, ..., x_I) \bigotimes S(y; y_0, ..., y_J)$. T est l'espace de dimension $(I+3)(J+3)$ des fonctions de la forme

$$u(x, y) = \sum_{p=0}^{I+2} \sum_{q=0}^{J+2} \beta_{pq} \Phi_p(x) \Psi_q(y), \tag{5}$$

Par construction, u est bicubique par morceaux et de classe C^2.

Définition 1 *Soit* $R_{ij} = \{(x, y) : x_{i-1} \le x \le x_i, y_{j-1} \le y \le y_j\}$. *Une fonction est polynomiale bicubique par morceaux si elle est définie sur chaque rectangle* R_{ij} *de la grille par une fonction polynomiale bicubique, i.e.*

$$u(x, y) = c_{ij}(x, y) = \sum_{p=0}^{3} \sum_{q=0}^{3} \alpha_{pq}^{ij} (x - x_{i-1})^p (y - y_{j-1})^q, \ (x, y) \in R_{ij}.$$

Théorème 2 *Si les valeurs suivantes sont données :*

$$\begin{cases} u_{ij} = u(x_i, y_j), & i = 0, ..., I; j = 0, ..., J, \\ p_{ij} = u_x(x_i, y_j), & i = 0, I; j = 0, ..., J, \\ q_{ij} = u_y(x_i, y_j), & i = 0, ..., I; j = 0, J, \\ s_{ij} = u_{xy}(x_i, y_j), & i = 0, I; j = 0, J, \end{cases} \tag{6}$$

alors il existe une unique fonction bicubique par morceaux de classe C^2 *qui vérifie (6).*

A.3 Evaluation de la fonction d'interpolation

Dans cette section, nous présentons une méthode d'évaluation de la fonction d'interpolation u en un point (x, y) quelconque. Rappelons que sur chaque rectangle R_{ij}, la fonction u est égale à la fonction polynomiale bicubique

$$c_{ij} = \sum_{p=0}^{3} \sum_{q=0}^{3} \gamma_{pq}^{ij} (x - x_{i-1})^p (y - y_{j-1})^q. \tag{7}$$

Lemme 1 *Si u_{ij}, p_{ij}, q_{ij} et s_{ij} sont connus aux quatre coins du rectangle R_{ij}, alors il existe une unique fonction polynomiale bicubique $c_{ij}(x,y)$ qui concorde avec ces valeurs. La matrice Λ_{ij} des coefficients γ_{mn}^{ij} dans (7) est donnée par l'équation matricielle suivante :*

$$\Lambda_{ij} \quad = \quad A(\Delta x_{i-1}) K_{ij} A(\Delta y_{j-1})^T, \tag{8}$$

où

$$K_{ij} = \begin{pmatrix} B_{i-1,j-1} & B_{i-1,j} \\ B_{i,j-1} & B_{ij} \end{pmatrix} \quad avec \quad B_{mn} = \begin{pmatrix} u_{mn} & q_{mn} \\ p_{mn} & s_{mn} \end{pmatrix},$$

et où la matrice $A(h)$ est définie par

$$A(h) \quad = \quad \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ -3/h^2 & -2/h & 3/h^2 & -1/h \\ 2/h^3 & 1/h^2 & -2/h^3 & 1/h^2 \end{pmatrix}.$$

Dérivées aux points de la grille

Lemme 2 *Si les valeurs (6) sont connues, alors les valeurs*

- $p_{ij} = u_x(x_i, y_j)$, *pour* $i = 1, ..., I-1; j = 0, ..., J$,
- $q_{ij} = u_y(x_i, y_j)$, *pour* $i = 0, ..., I; j = 1, ..., J-1$,
- $s_{ij} = u_{xy}(x_i, y_j)$, *pour* $i = 1, ..., I-1; j = 0, J$ *et* $i = 0, ..., I; j = 1, ..., J-1$,

sont déterminées de manière unique par les $2I + J + 5$ systèmes linéaires qui comptent en tout $3IJ + I + J - 5$ équations. Pour $j = 0, ..., J$,

$$\begin{aligned} &\Delta x_{i-1} p_{i+1,j} + 2(\Delta x_{i-1} + \Delta x_i) p_{ij} + \Delta x_i p_{i-1,j} = \\ &\qquad 3\left[\frac{\Delta x_{i-1}}{\Delta x_i}(u_{i+1,j} - u_{i,j}) + \frac{\Delta x_i}{\Delta x_{i-1}}(u_{i,j} - u_{i-1,j})\right], \ i = 1, ..., I-1; \end{aligned} \tag{9}$$

pour $i = 0, ..., I$,

$$\begin{aligned} &\Delta y_{j-1} q_{i,j+1} + 2(\Delta y_{j-1} + \Delta y_j) q_{ij} + \Delta y_j q_{i,j-1} = \\ &\qquad 3\left[\frac{\Delta y_{j-1}}{\Delta y_j}(u_{i,j+1} - u_{i,j}) + \frac{\Delta y_j}{\Delta y_{j-1}}(u_{i,j} - u_{i,j-1})\right], \ j = 1, ..., J-1; \end{aligned} \tag{10}$$

pour $j = 0, J$,

$$\begin{aligned} &\Delta x_{i-1} s_{i+1,j} + 2(\Delta x_{i-1} + \Delta x_i) s_{ij} + \Delta x_i s_{i-1,j} = \\ &\qquad 3\left[\frac{\Delta x_{i-1}}{\Delta x_i}(q_{i+1,j} - q_{i,j}) + \frac{\Delta x_i}{\Delta x_{i-1}}(q_{i,j} - q_{i-1,j})\right], \ i = 1, ..., I-1; \end{aligned} \tag{11}$$

pour $i = 0, ..., I$,

$$\Delta y_{j-1} s_{i,j+1} + 2(\Delta y_{j-1} + \Delta y_j) s_{ij} + \Delta y_j s_{i,j-1} = 3\left[\frac{\Delta y_{j-1}}{\Delta y_j}(p_{i,j+1} - p_{i,j}) + \frac{\Delta y_j}{\Delta y_{j-1}}(p_{i,j} - p_{i,j-1})\right], \; j = 1, ..., J-1; \tag{12}$$

Algorithme d'évaluation de $u(x, y)$

Etape 1 Calculer les valeurs p_{ij}, q_{ij} et s_{ij} pour tous les points de la grille grâce aux équations précédentes en utilisant l'élimination de Gauss (les matrices des $2I + J - 5$ systèmes sont tridiagonales et à diagonale strictement dominante).

Etape 2 Une fois que l'on dispose des valeurs u, u_x, u_y et u_{xy} en tout point de la grille, calculer dans chaque rectangle R_{ij} les coefficients γ_{pq}^{ij} de la fonction polynomiale bicubique dans ce rectangle en utilisant (8).

Evaluer la fonction d'interpolation en un point (x, y) revient alors à trouver les indices (i, j) tels que $(x, y) \in R_{ij}$, puis à évaluer la fonction polynomiale bicubique correspondante.

A.4 Application dans l'algorithme de calibration

Pour résoudre l'EDP de Dupire, il faut connître la volatilité σ en tout point de la grille fine de différences finies de taille $N \times M$. La volatilité étant définie comme une spline bicubique sur une grille grossière de taille $n \times m$, nous procédons de façon suivante pour obtenir la valeur de σ en un point quelconque de la grille fine :

- on calcule la fonction d'interpolation par spline bicubique sur tout l'espace $[y_{min}; y_{max}] \times [t_0; T_{max}]$ à partir des degrés de libertés sur la grille grossière $n \times m$, en appliquant l'algorithme décrit précédemment ;
- on évalue la fonction d'interpolation au point considéré de la grille fine $N \times M$.

B Calcul du gradient de F_1

On décrit dans cette annexe le calcul du gradient de la fonction

$$F_1(\sigma) = \| \nabla\sigma \|^2.$$

Pour éviter toute confusion, nous notons Σ l'ensemble des degrés de liberté de la volatilité et σ la fonction de deux variables associée.

B.1 Première approche

Regardons dans un premier temps le calcul du gradient de F_1.

$$\begin{aligned}
F_1(\Sigma) &= \| \nabla\sigma \|^2 \\
&= \iint_R \left[\left(\frac{\partial\sigma}{\partial y}(y,t) \right)^2 + \left(\frac{\partial\sigma}{\partial t}(y,t) \right)^2 \right] dy\, dt \\
&= \sum_{i=1}^{n} \sum_{j=1}^{m} \iint_{R_{ij}} \left[\left(\sum_{p=1}^{3} \sum_{q=0}^{3} p\gamma_{pq}^{ij} (y - y_{i-1})^{p-1} (t - t_{j-1})^q \right)^2 \right. \\
&\qquad \left. + \left(\sum_{p=0}^{3} \sum_{q=1}^{3} q\gamma_{pq}^{ij} (y - y_{i-1})^{p} (t - t_{j-1})^{q-1} \right)^2 \right] dy\, dt \\
&= \sum_{i=1}^{n} \sum_{j=1}^{m} \left(\int_{y_{i-1}}^{y_i} \int_{t_{j-1}}^{t_j} A(y,t)\, dy\, dt + \int_{y_{i-1}}^{y_i} \int_{t_{j-1}}^{t_j} B(y,t)\, dy\, dt \right)
\end{aligned}$$

où

$$\begin{aligned}
A(y,t) &= \left(\sum_{p=1}^{3} \sum_{q=0}^{3} p\gamma_{pq}^{ij} (y - y_{i-1})^{p-1} (t - t_{j-1})^q \right)^2, \\
B(y,t) &= \left(\sum_{p=0}^{3} \sum_{q=1}^{3} q\gamma_{pq}^{ij} (y - y_{i-1})^{p} (t - t_{j-1})^{q-1} \right)^2.
\end{aligned}$$

Après calculs, on obtient une expression de F_1 comme fonction quadratique des γ^{ij} :

$$F_1(\Sigma) = \sum_{i,j} (\gamma^{ij})^T Q^{ij} \gamma^{ij}$$

où γ_{ij} est le vecteur formé par les coefficients γ_{pq}^{ij}. Puis, grâce aux sytèmes d'équations (9) à (12) et à l'équation (8), on peut exprimer les γ^{ij} linéairement par rapport au vecteur de paramètres Σ :

$$\gamma^{ij} = M^{ij} \cdot \Sigma.$$

On obtient finalement :

$$F_1(\Sigma) \quad = \quad \sum_{i,j} \Sigma^T [(M^{ij})^T Q^{ij} M^{ij}] \Sigma.$$

Mais tous ces calculs demandent un travail vraiment important : nous présentons ci-dessous une solution alternative beaucoup simple à mettre en oeuvre.

B.2 Une expression simplifiée du problème

Au lieu de pénaliser le gradient dans la fonction objectif, on choisit de pénaliser la somme des carrés des dérivées premières calculées aux points de la grille :

$$F(\sigma) \quad = \quad G(\sigma) + \lambda \sum_{i,j} (\sigma_{y,ij}^2 + \sigma_{t,ij}^2)$$

avec, pour tout point (y_i, T_j) de la grille :

$$\sigma_{y,ij} = \frac{\partial \sigma}{\partial y}(y_i, T_j) \qquad\qquad \sigma_{t,ij} = \frac{\partial \sigma}{\partial T}(y_i, T_j).$$

On note

$$F_1(\Sigma) = \sum_{i,j} (\sigma_{y,ij}^2 + \sigma_{t,ij}^2).$$

On peut écrire

$$F_1(\Sigma) \quad = \quad \sum_{i,j} \sigma_{\cdot,ij}^T \sigma_{\cdot,ij}.$$

D'autre part, grâce aux systèmes d'équations (9) à (12), on peut exprimer ces dérivées linéairement en fonction du vecteur de paramètres Σ :

$$\sigma_{\cdot,ij} = N^{ij} \cdot \Sigma.$$

Remarque 2 *N^{ij} est une matrice de taille* $2 \times ((n+1)(m+1) + 2(n+1) + 2(m+1) + 4) = 2(n+3)(m+3)$.

On a donc :

$$\begin{aligned} F_1(\Sigma) &= \sum_{i,j} (N^{ij} \cdot \Sigma)^T \cdot (N^{ij} \cdot \Sigma) \\ &= \sum_{i,j} \Sigma^T (N^{ij})^T N^{ij} \cdot \Sigma, \end{aligned}$$

et en dérivant :

$$\nabla F_1(\Sigma) \quad = \quad \sum_{i,j} (N^{ij})^T N^{ij} \cdot \Sigma. \tag{13}$$

Le calcul du gradient de F_1 se résume donc au calcul des matrices N^{ij}.

Calcul de la matrice N^{ij}

Ecrivons les systèmes (9) et (10) pour la fonction $\sigma(y,t)$ et ses dérivées premières. Pour $j = 0, ..., m$,

$$\Delta y_{i-1}\sigma_{y,i+1,j} + 2(\Delta y_{i-1} + \Delta y_i)\sigma_{y,ij} + \Delta y_i \sigma_{y,i-1,j} = 3\left[\frac{\Delta y_{i-1}}{\Delta y_i}(\sigma_{i+1,j} - \sigma_{i,j}) + \frac{\Delta y_i}{\Delta y_{i-1}}(\sigma_{i,j} - \sigma_{i-1,j})\right],\ i = 1, ..., n-1; \tag{14}$$

pour $i = 0, ..., n$,

$$\Delta t_{j-1}\sigma_{t,i,j+1} + 2(\Delta t_{j-1} + \Delta t_j)\sigma_{t,ij} + \Delta t_j \sigma_{t,i,j-1} = 3\left[\frac{\Delta t_{j-1}}{\Delta t_j}(\sigma_{i,j+1} - \sigma_{i,j}) + \frac{\Delta t_j}{\Delta t_{j-1}}(\sigma_{i,j} - \sigma_{i,j-1})\right],\ j = 1, ..., m-1. \tag{15}$$

En y ajoutant les conditions aux bords, (14) peut s'écrire sous la forme matricielle :

$$A\sigma_{y,\cdot j} + \Delta y_1 \sigma_{y,0j} e_1 + \Delta y_{n-1}\sigma_{y,nj} e^{n-1} = 3B\sigma_{\cdot j} - 3\frac{\Delta y_1}{\Delta y_0}\sigma_{0j}e_1 + 3\frac{\Delta y_{n-2}}{\Delta y_{n-1}}\sigma_{nj}e_{n-1} \tag{16}$$

$$A = \begin{pmatrix} (\Delta y_0 + \Delta y_1) & \Delta y_0 & 0 & \dots & \dots & 0 \\ \Delta y_2 & (\Delta y_1 + \Delta y_2) & \Delta y_1 & 0 & \dots & 0 \\ 0 & \ddots & \ddots & \ddots & \ddots & \vdots \\ \vdots & \ddots & \ddots & \ddots & \ddots & 0 \\ \vdots & & \ddots & \ddots & \ddots & \Delta y_{n-3} \\ 0 & \dots & \dots & 0 & \Delta y_{n-1} & (\Delta y_{n-2} + \Delta y_{n-1}) \end{pmatrix},$$

$$B = \begin{pmatrix} \left(\frac{\Delta y_1}{\Delta y_0} - \frac{\Delta y_0}{\Delta y_1}\right) & \frac{\Delta y_0}{\Delta y_1} & 0 & \dots & \dots & 0 \\ -\frac{\Delta y_2}{\Delta y_1} & \left(\frac{\Delta y_2}{\Delta y_1} - \frac{\Delta y_1}{\Delta y_2}\right) & \frac{\Delta y_1}{\Delta y_2} & 0 & \dots & 0 \\ 0 & \ddots & \ddots & \ddots & \ddots & \vdots \\ \vdots & \ddots & \ddots & \ddots & \ddots & 0 \\ \vdots & & \ddots & \ddots & \ddots & \frac{\Delta y_{n-3}}{\Delta y_{n-2}} \\ 0 & \dots & \dots & 0 & -\frac{\Delta y_{n-1}}{\Delta y_{n-2}} & \left(\frac{\Delta y_{n-1}}{\Delta y_{n-2}} - \frac{\Delta y_{n-2}}{\Delta y_{n-1}}\right) \end{pmatrix}$$

et

$$e_1 = \begin{pmatrix} 1 \\ 0 \\ \vdots \\ 0 \end{pmatrix}, \qquad e_{n-1} = \begin{pmatrix} 0 \\ \vdots \\ 0 \\ 1 \end{pmatrix}.$$

Remarque 3 *A et B sont des matrices carrées de taille $n-1$; e_1 et e_{n-1} sont des vecteurs de taille $n-1$.*

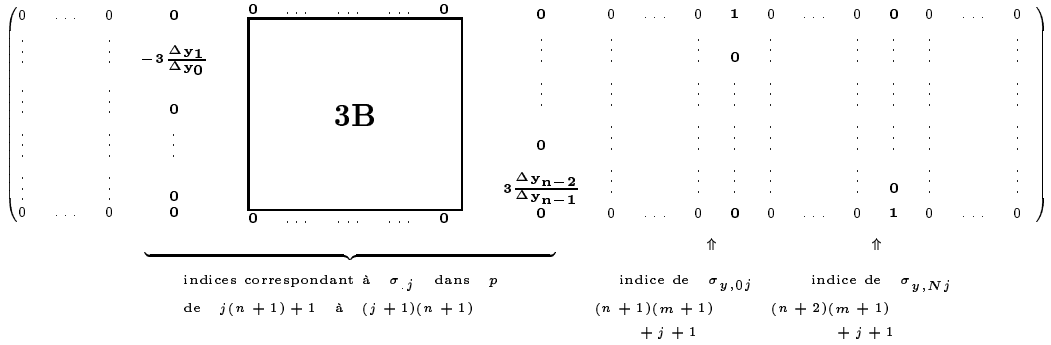
On cherche à présent à exprimer $\Sigma_{y,\cdot j}$ linéairement en fonction du vecteur de paramètres P. Rappelons que le vecteur P de taille $(n+3)(m+3)$ contient les données permettant de définir la fonction d'interpolation σ :

$$P = \begin{pmatrix} \Sigma_{\cdot 0} \\ \vdots \\ \Sigma_{\cdot m} \\ \Sigma_{y,0\cdot} \\ \Sigma_{y,n\cdot} \\ \Sigma_{t,\cdot 0} \\ \Sigma_{t,\cdot m} \\ \sigma_{yt,00} \\ \sigma_{yt,n0} \\ \sigma_{yt,0m} \\ \sigma_{yt,nm} \end{pmatrix}.$$

On construit les matrices $\mathcal{A}$ et $\mathcal{B}_j$ à partir de A et B :

$$\mathcal{A} = \begin{pmatrix} 1 & 0 & \dots & 0 & 0 \\ \Delta y_1 & & & & 0 \\ 0 & & A & & \vdots \\ \vdots & & & & 0 \\ 0 & & & & \Delta y_{n-1} \\ 0 & 0 & \dots & 0 & 1 \end{pmatrix}$$

et on définit $\mathcal{B}_j$ par la matrice :



On a alors :

$$\mathcal{A}\Sigma_{y,\cdot j} \quad = \quad \mathcal{B}_j p. \tag{17}$$

De la même manière, on construit une équation matricielle à partir de (15). De (15), on déduit :

$$\begin{aligned} C\sigma_{t,i\cdot} + \Delta t_1 \sigma_{t,i0} f_1 + \Delta t_{m-1} \sigma_{t,im} f_{m-1} = \\ 3D\sigma_{i\cdot} - 3\frac{\Delta t_1}{\Delta t_0}\sigma_{i0} f_1 + 3\frac{\Delta t_{m-2}}{\Delta t_{m-1}}\sigma_{im} f_{m-1} \end{aligned} \tag{18}$$

où

$$C \quad = \quad \begin{pmatrix} (\Delta t_0 + \Delta t_1) & \Delta t_0 & 0 & \dots & \dots & 0 \\ \Delta t_2 & (\Delta t_1 + \Delta t_2) & \Delta t_1 & 0 & \dots & 0 \\ 0 & \ddots & \ddots & \ddots & \ddots & \vdots \\ \vdots & \ddots & \ddots & \ddots & \ddots & 0 \\ \vdots & & \ddots & \ddots & \ddots & \Delta t_{m-3} \\ 0 & \dots & \dots & 0 & \Delta t_{m-1} & (\Delta t_{m-2} + \Delta t_{m-1}) \end{pmatrix},$$

$$D \quad = \quad \begin{pmatrix} \left(\frac{\Delta t_1}{\Delta t_0} - \frac{\Delta t_0}{\Delta t_1}\right) & \frac{\Delta t_0}{\Delta t_1} & 0 & \dots & \dots & 0 \\ -\frac{\Delta t_2}{\Delta t_1} & \left(\frac{\Delta t_2}{\Delta t_1} - \frac{\Delta t_1}{\Delta t_2}\right) & \frac{\Delta t_1}{\Delta t_2} & 0 & \dots & 0 \\ 0 & \ddots & \ddots & \ddots & \ddots & \vdots \\ \vdots & \ddots & \ddots & \ddots & \ddots & 0 \\ \vdots & & \ddots & \ddots & \ddots & \frac{\Delta t_{m-3}}{\Delta t_{m-2}} \\ 0 & \dots & \dots & 0 & -\frac{\Delta t_{m-1}}{\Delta t_{m-2}} & \left(\frac{\Delta t_{m-1}}{\Delta t_{m-2}} - \frac{\Delta t_{m-2}}{\Delta t_{m-1}}\right) \end{pmatrix}$$

et

$$f_1 = \begin{pmatrix} 1 \\ 0 \\ \vdots \\ 0 \end{pmatrix}, \qquad f_{m-1} = \begin{pmatrix} 0 \\ \vdots \\ 0 \\ 1 \end{pmatrix}.$$

Remarque 4 *C et D sont des matrices carrées de taille $m-1$; f_1 et f_{m-1} sont des vecteurs de taille $m-1$.*

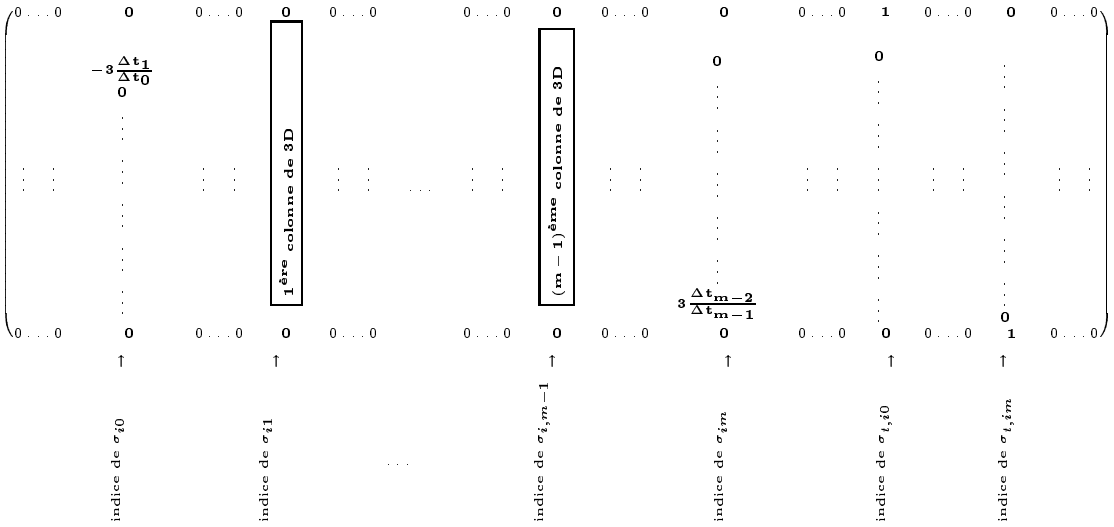
On cherche à présent à exprimer $\Sigma_{t,i.}$ linéairement en fonction du vecteur de paramètres P grâce à l'équation

$$\mathcal{C}\Sigma_{t,i.} \quad = \quad \mathcal{D}_i P. \tag{19}$$

Pour obtenir cette équation il faut définir $\mathcal{C}$ par

$$\mathcal{C} \quad = \quad \begin{pmatrix} 1 & 0 \ \ldots \ 0 & 0 \\ \Delta t_1 & & 0 \\ 0 & \boxed{\mathrm{C}} & \vdots \\ \vdots & & 0 \\ 0 & & \Delta t_{m-1} \\ 0 & 0 \ \ldots \ 0 & 1 \end{pmatrix}$$

et $\mathcal{D}_i$ par la matrice



Grâce à (17) et (19) et après le réordonnancement suivant :

$$N^{ij} = \begin{pmatrix} i^{\text{ème}} \text{ ligne de } \mathcal{A}^{-1}\mathcal{B}_i \\ j^{\text{ème}} \text{ ligne de } \mathcal{C}^{-1}\mathcal{D}_i \end{pmatrix},$$

on peut construire les matrices N^{ij} telles que :

$$\sigma_{.,ij} = \begin{pmatrix} \sigma_{y,ij} \\ \sigma_{t,ij} \end{pmatrix} = N^{ij} \cdot p. \tag{20}$$

Remarque 5 *La simple pénalisation des dérivées du premier ordre n'empêche pas l'apparition de "pics" à l'intérieur des rectangles R_{ij}. Pour éviter ce phénomène, on pourra par la suite pénaliser aussi toutes les dérivées du second ordre. La nouvelle fonction F_1 serait alors*

$$F_1(\sigma) = \sum_{i,j} (\sigma^2_{y,ij} + \sigma^2_{t,ij} + \sigma^2_{yt,ij} + \sigma^2_{yy,ij} + \sigma^2_{tt,ij}).$$

C Calcul du gradient de G

On décrit dans cette annexe le calcul du gradient de la fonction

$$G(\sigma) = \frac{1}{2}\|V(\sigma) - \tilde{V}\|^2$$

qui mesure l'écart entre les données simulées en résolvant l'EDP de Dupire et les données du marché. Pour calculer ce gradient de façon analytique, nous utilisons la méthode de l'état adjoint.

Pour éviter toute confusion, nous notons Σ l'ensemble des degrés de liberté de la volatilité et σ la fonction de deux variables associée. De plus, nous écrivons G sous la forme

$$G(\Sigma) = \tfrac{1}{2}\|\varphi(U(\Sigma)) - \tilde{V}\|_2^2,$$

avec $U \in I\!R^{n_U}$ le vecteur d'état

$$U = (u^0, ..., u^M)^T$$

où les $u^j = (u_1^j ... u_{N-1}^j)^T \in I\!R^{N-1} \quad \forall j = 0, ..., M$ sont calculés par le système d'équations (équations d'état) suivant :

$$(E) \begin{cases} f - u^0 = 0, \\ b^j - H^j u^{j+1} = 0 \quad j = 0, ..., M-1. \end{cases}$$

avec, pour $j = 0, ..., M-1$,

$$\begin{aligned} b^j &= (I - \theta k \hat{A}^j) u^j - S_0 k \left(\theta \alpha_{1,T_j} e^{-q(T_j - t_0)} + (1-\theta) \alpha_{1,T_{j+1}} e^{-q(T_{j+1} - t_0)} \right) e_1^{N-1} \\ &\quad \in I\!R^{N-1}, \\ H^j &= I + (1-\theta) k \hat{A}^{j+1} \quad \in \mathcal{M}_{N-1,N-1}. \end{aligned}$$

On définit le lagrangien par :

$$\begin{aligned} \mathcal{L} : I\!R^{n_U} \times I\!R^{n_U} \times I\!R^{n_\Sigma} &\to I\!R \\ (U, \bar{U}, \Sigma) &\to \mathcal{L}(U, \bar{U}, \Sigma) \end{aligned}$$

avec

$$\mathcal{L}(U, \bar{U}, \Sigma) = \tfrac{1}{2}\|\varphi(U) - \tilde{V}\|_2^2 + < f - u^0, \bar{u}^0 >_{N-1} + \sum_{j=0}^{M-1} < b^j - H^j u^{j+1}, \bar{u}^{j+1} >_{N-1} . \quad (21)$$

Remarque 6 *b^j dépend de u^j et de Σ, tandis que H^j ne dépend que de Σ.*

Pour un Σ fixé, on choisit U solution de l'équation d'état (E) correspondante. On a alors l'égalité suivante :

$$G(\Sigma) \quad = \quad \mathcal{L}(U, \bar{U}, \Sigma).$$

En différentiant G, on obtient :

$$\delta G \quad = \quad \frac{\partial \mathcal{L}}{\partial U}(U, \bar{U}, \Sigma)\delta U + \frac{\partial \mathcal{L}}{\partial \Sigma}(U, \bar{U}, \Sigma)\delta\Sigma \quad \forall \quad \delta U \in I\!R^{n_U}$$

car, U étant solution de (E),

$$\frac{\partial \mathcal{L}}{\partial \bar{U}}(U, \bar{U}, \Sigma)\delta U \quad = \quad 0.$$

On choisit alors $\bar{U}$ tel que

$$\frac{\partial \mathcal{L}}{\partial U}(U, \bar{U}, \Sigma)\delta U = 0 \quad \forall \quad \delta U \in I\!R^{n_U},$$

c'est-à-dire

$$\nabla_U \mathcal{L}(U, \bar{U}, \Sigma) \quad = \quad \mathbf{0}. \tag{22}$$

Pour Σ et U fixés, cela revient à résoudre un système linéaire admettant comme unique solution l'état adjoint $\bar{U}$. Au point $(U, \bar{U}, \Sigma)$ ainsi défini, le gradient de G se réduit à :

$$\nabla_\Sigma G = \nabla_\Sigma \mathcal{L}. \tag{23}$$

Equation de l'état adjoint

Réécrivons le système (22) :

$$\begin{cases} \nabla_{u^0}\mathcal{L} = \left[\dfrac{\partial \varphi}{\partial u^0}(U)\right]^T (\varphi(U) - \tilde{V}) - \bar{u}^0 + \left[\dfrac{\partial b^0}{\partial u^0}\right]^T \bar{u}^1 = 0, \\ \nabla_{u^j}\mathcal{L} = \left[\dfrac{\partial \varphi}{\partial u^j}(U)\right]^T (\varphi(U) - \tilde{V}) - (H^{j-1})^T \bar{u}^j + \left[\dfrac{\partial b^j}{\partial u^j}\right]^T \bar{u}^{j+1} = 0 \quad j = 1...M-1, \\ \nabla_{u^M}\mathcal{L} = \left[\dfrac{\partial \varphi}{\partial u^M}(U)\right]^T (\varphi(U) - \tilde{V}) - (H^{M-1})^T \bar{u}^M = 0. \end{cases}$$

Ce systéme est équivalent à :

$$\begin{cases} (H^{M-1})^T \bar{u}^M = \left[\dfrac{\partial \varphi}{\partial u^M}(U)\right]^T (\varphi(U) - \tilde{V}), \\ (H^{j-1})^T \bar{u}^j = \left[\dfrac{\partial \varphi}{\partial u^j}(U)\right]^T (\varphi(U) - \tilde{V}) + \left[\dfrac{\partial b^j}{\partial u^j}\right]^T \bar{u}^{j+1} \quad j = M-1, ..., 1, \\ \bar{u}^0 = \left[\dfrac{\partial \varphi}{\partial u^0}(U)\right]^T (\varphi(U) - \tilde{V}) + \left[\dfrac{\partial b^0}{\partial u^0}\right]^T \bar{u}^1. \end{cases} \tag{24}$$

Pour calculer l'état adjoint $\bar{U}$, il faut déterminer les dérivées partielles $\frac{\partial b^j}{\partial u^j}$ et $\frac{\partial \varphi}{\partial u^j}(U)$. On a de manière immédiate

$$\frac{\partial b^j}{\partial u^j} = (I - \theta k \hat{A}^j) \quad \in \mathcal{M}_{N-1,N-1}.$$

Précisons la définition de la fonction d'observation $\varphi : I\!R^{n_U} \to I\!R^{n_d}$. A partir des valeurs des options aux points de la grille $u_k^l = U(y_k, T_l)$ on cherche à déterminer le prix d'une option dont le couple $(y = \log(K), T)$ n'est pas forcément sur la grille. On évalue le prix de cette option par interpolation bilinéaire des 4 valeurs qui l'entourent sur la grille fine. Supposons que l'on veuille calculer le prix de l'option dont le couple strike-maturité (y, T) est tel que :

$$\begin{aligned} y_k \leq y < y_{k+1}, \\ T_l \leq T < T_{l+1}. \end{aligned}$$

Calculons dans un premier temps par interpolation linéaire $U(y, T_l)$ et $U(y, T_{l+1})$:

$$\begin{aligned} U(y, T_l) &= \left(1 - \frac{y_{k+1} - y}{y_{k+1} - y_k}\right) u_{k+1}^l + \frac{y_{k+1} - y}{y_{k+1} - y_k} u_k^l, \\ U(y, T_{l+1}) &= \left(1 - \frac{y_{k+1} - y}{y_{k+1} - y_k}\right) u_{k+1}^{l+1} + \frac{y_{k+1} - y}{y_{k+1} - y_k} u_k^{l+1}. \end{aligned}$$

Puis on fait une interpolation linéraire entre $U(y, T_l)$ et $U(y, T_{l+1})$ pour obtenir $U(y, T)$:

$$U(y, T) = \left(1 - \frac{T_{l+1} - T}{T_{l+1} - T_l}\right) U(y, T_{l+1}) + \frac{T_{l+1} - T}{T_{l+1} - T_l} U(y, T_l).$$

En posant

$$\begin{aligned} a_l &= \frac{T_{l+1} - T}{T_{l+1} - T_l}, \\ b_k &= \frac{y_{k+1} - y}{y_{k+1} - y_k}, \end{aligned}$$

on obtient finalement

$$U(y, T) = \varphi_{k,l}(U) = (1 - a_l)[(1 - b_k) u_{k+1}^{l+1} + b_k u_k^{l+1}] + a_l[(1 - b_k) u_{k+1}^l + b_k u_k^l].$$

Soient $(y^d, T^d)_{1 \leq d \leq n_d}$ l'ensemble des couples strike/maturité pour lesquels on connait le prix du marché. La fonction d'évaluation φ est alors définie de la manière suivante :

$$\begin{aligned} \varphi : I\!R^{n_U} &\to I\!R^{n_d} \\ U &\to \varphi(U) = \left(U(y^d, T^d)\right)_{d=1 \ldots n_d} \end{aligned}$$

avec

$$U(y^d, T^d) = \varphi_{kl}(U) \text{ avec } y_k \leq y^d < y_{k+1} \text{ et } T_l \leq T^d < T_{l+1}.$$

La matrice $\dfrac{\partial \varphi}{\partial u^j}(U) \in \mathcal{M}_{n_d, N-1}$ est donc définie par :

$$\frac{\partial \varphi}{\partial u^j}(U) \quad = \quad \left(\frac{\partial \varphi_{kl}}{\partial u_p^j}(U) \right)_{\substack{d=1,\dots,n_d \\ p=1,\dots,N-1}}$$

où k et l sont définis pour chaque d par $y_k \leq y^d < y_{k+1}$ et $T_l \leq T^d < T_{l+1}$, et où

$$\frac{\partial \varphi_{kl}}{\partial u_p^j}(U) \quad = \quad \begin{cases} a_l b_k & \text{si } p = k \text{ et } j = l, \\ (1-a_l) b_k & \text{si } p = k \text{ et } j = l+1, \\ a_l (1-b_k) & \text{si } p = k+1 \text{ et } j = l, \\ (1-a_l)(1-b_k) & \text{si } p = k+1 \text{ et } j = l+1, \\ 0 & \text{sinon.} \end{cases}$$

Calcul de ∇G

On est à présent en mesure de calculer la solution $\bar{U}$ de l'équation de l'état adjoint (A). Une fois Σ, U et $\bar{U}$ connus, il reste à calculer

$$\nabla_\Sigma G \quad = \quad \nabla_\Sigma \mathcal{L}(U, \bar{U}, \Sigma) \tag{25}$$

$$\quad = \quad \sum_{j=0}^{M-1} \left[\left(\frac{\partial b^j}{\partial \Sigma} \right)^T - \frac{\partial (H^j u^{j+1})}{\partial \Sigma} \right)^T \bar{u}^{j+1} \Bigg]. \tag{26}$$

Remarquons que ce calcul se ramène à celui de $\dfrac{\partial (\hat{A}^j u^j)}{\partial \Sigma}$:

$$\frac{\partial b^j}{\partial \Sigma} \quad = \quad -\theta k \frac{\partial (\hat{A}^j u^j)}{\partial \Sigma} - S_0 k \left(\theta \frac{\partial \alpha_{1,T_j}}{\partial \Sigma} e^{-q(T_j - t_0)} + (1-\theta) \frac{\partial \alpha_{1,T_{j+1}}}{\partial \Sigma} e^{-q(T_{j+1} - t_0)} \right) e_1^{N-1}, \tag{27}$$

$$\frac{\partial H^j u^{j+1}}{\partial \Sigma} \quad = \quad (1-\theta) k \frac{\partial (\hat{A}^{j+1} u^{j+1})}{\partial \Sigma}. \tag{28}$$

Or

$$\hat{A}^j u^j \quad = \quad \begin{pmatrix} \beta_{1,T_j} u_1^j + \gamma_{1,T_j} u_2^j \\ \vdots \\ \alpha_{i,T_j} u_{i-1}^j + \beta_{i,T_j} u_i^j + \gamma_{i,T_j} u_{i+1}^j \\ \vdots \\ \alpha_{N-1,T_j} u_{N-2}^j + \beta_{N-1,T_j} u_{N-1}^j \end{pmatrix} \quad \in I\!R^{N-1},$$

donc

$$\frac{\partial \hat{A}^j u^j}{\partial \Sigma} = \begin{pmatrix} \frac{\partial \beta_{1,T_j}}{\partial \Sigma} u_1^j + \frac{\partial \gamma_{1,T_j}}{\partial \Sigma} u_2^j \\ \vdots \\ \frac{\partial \alpha_{i,T_j}}{\partial \Sigma} u_{i-1}^j + \frac{\partial \beta_{i,T_j}}{\partial \Sigma} u_i^j + \frac{\partial \gamma_{i,T_j}}{\partial \Sigma} u_{i+1}^j \\ \vdots \\ \frac{\partial \alpha_{N-1,T_j}}{\partial \Sigma} u_{N-2}^j + \frac{\partial \beta_{N-1,T_j}}{\partial \Sigma} u_{N-1}^j \end{pmatrix} \in \mathcal{M}_{N-1,n_\Sigma}, \tag{29}$$

avec

$$\begin{aligned} \frac{\partial \alpha_{i,T_j}}{\partial \Sigma} &= \left[-\frac{2}{(h_i + h_{i-1})h_{i-1}} - \frac{1}{2h_{i-1}} \right] \sigma(y_i, T_j) \frac{\partial \sigma(y_i, T_j)}{\partial \Sigma}, \\ \frac{\partial \beta_{i,T_j}}{\partial \Sigma} &= \left[\frac{2}{h_i h_{i-1}} + \frac{1}{2} \left(\frac{1}{h_{i-1}} - \frac{1}{h_i} \right) \right] \sigma(y_i, T_j) \frac{\partial \sigma(y_i, T_j)}{\partial \Sigma}, \\ \frac{\partial \gamma_{i,T_j}}{\partial \Sigma} &= \left[-\frac{2}{(h_i + h_{i-1})h_i} + \frac{1}{2h_i} \right] \sigma(y_i, T_j) \frac{\partial \sigma(y_i, T_j)}{\partial \Sigma}. \end{aligned}$$

Il faut donc évaluer la fonction σ aux points (y_i, T_j), grâce à une interpolation par spline bicubique des valeurs de σ sur la grille grossière :

$$\sigma(y_i, T_j) = \sum_{p=0}^{3} \sum_{q=0}^{3} c_{pq}^{kl} (y - y_{k-1})^p (T - T_{l-1})^q \quad \text{pour } (y_i, T_j) \in R_{kl}.$$

Grâce à l'égalité matricielle (8), on peut exprimer c_{pq}^{kl} en fonction des valeurs de σ et de ses dérivées aux différents points de la grille :

$$\begin{aligned} c_{pq}^{kl} &= \sum_{r=0}^{n} \sum_{s=0}^{m} \left(a_{pq,rs}^{kl} \sigma_{kl} + b_{pq,rs}^{kl} \sigma_{y,kl} + d_{pq,rs}^{kl} \sigma_{T,kl} + e_{pq,rs}^{kl} \sigma_{yT,kl} \right) \\ &= (\mathcal{U}_{pq}^{kl})^T \cdot S, \end{aligned}$$

ce qui nous permet d'écrire $\sigma(y_i, T_j)$ sous la forme suivante (pour $(y_i, T_j) \in R_{kl}$) :

$$\begin{aligned} \sigma(y_i, T_j) &= \sum_{p=0}^{3} \sum_{q=0}^{3} (y - y_{k-1})^p (T - T_{l-1})^q (\mathcal{U}_{pq}^{kl})^T \cdot S, \\ &= \sum_{p=0}^{3} \sum_{q=0}^{3} (y - y_{k-1})^p (T - T_{l-1})^q (\mathcal{U}_{pq}^{kl})^T \cdot R \cdot \Sigma, \end{aligned}$$

soit

$$\frac{\partial \sigma(y_i, T_j)}{\partial \Sigma} = \sum_{p=0}^{3} \sum_{q=0}^{3} \left[(y - y_{k-1})^p (T - T_{l-1})^q (\mathcal{U}_{pq}^{kl})^T \right] \cdot R. \tag{30}$$

avec, en utilisant les même notations que dans la partie B :

$$S = \begin{pmatrix} \sigma_{\cdot 0} \\ \vdots \\ \sigma_{\cdot m} \\ \sigma_{y,\cdot 0} \\ \vdots \\ \sigma_{y,\cdot m} \\ \sigma_{T,\cdot 0} \\ \vdots \\ \sigma_{T,\cdot m} \\ \sigma_{yT,\cdot 0} \\ \vdots \\ \sigma_{yT,\cdot m} \end{pmatrix} \quad \text{et} \quad \mathcal{U}_{pq}^{kl} = \begin{pmatrix} a_{pq,\cdot 0}^{kl} \\ \vdots \\ a_{pq,\cdot m}^{kl} \\ b_{pq,\cdot 0}^{kl} \\ \vdots \\ b_{pq,\cdot m}^{kl} \\ d_{pq,\cdot 0}^{kl} \\ \vdots \\ d_{pq,\cdot m}^{kl} \\ e_{pq,\cdot 0}^{kl} \\ \vdots \\ e_{pq,\cdot m}^{kl} \end{pmatrix} \in \mathcal{M}_{4(m+1)(n+1),1}.$$

Il faut donc à présent déterminer la matrice $R \in \mathcal{M}_{4(n+1)(m+1),n_\Sigma}$ qui lie S à Σ par

$$S = R \cdot \Sigma \quad \in I\!R^{4(n+1)(m+1)}.$$

Construction de la matrice R

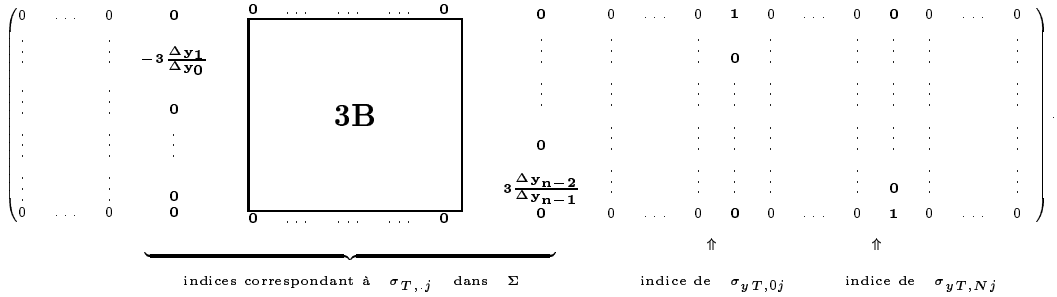
Ce calcul a déjà été fait en partie lors du calcul de ∇F_1. En effet, rappelons que :

$$\begin{aligned} \sigma_{T,i\cdot} &= \mathcal{C}^{-1}\mathcal{D}_i \cdot \Sigma \quad \forall i, \\ \sigma_{y,\cdot j} &= \mathcal{A}^{-1}\mathcal{B}_j \cdot \Sigma \quad \forall j. \end{aligned}$$

Après un réordonnancement, on obtient facilement les $3(n+1)(m+1)$ premières colonnes de R. La dépendance des dérivées croisées en Σ est plus délicate à obtenir. On procède en deux étapes : les dérivées croisées sont dans un premier temps calculées au bord, puis on en déduit par un second système les valeurs aux points intérieurs de la grille. Le système d'équations (11) peut s'écrire sous la forme matricielle suivante :

$$\begin{aligned} &A\sigma_{yT,\cdot j} + \Delta y_1 \sigma_{yT,0j} e_1 + \Delta y_{n-1} \sigma_{yT,nj} e_{n-1} \\ &\qquad = 3B\sigma_{T,\cdot j} - 3\frac{\Delta y_1}{\Delta y_0}\sigma_{T,0j}e_1 + 3\frac{\Delta y_{n-2}}{\Delta y_{n-1}}\sigma_{T,nj}e_{n-1} \quad \text{pour } j = 0, m \end{aligned}$$

où e_1,e_{N-1}, A, B sont définis comme précédemment (cf. section B). On définit $\mathcal{H}_j$ par la matrice :



Alors le système s'écrit

$$\sigma_{yT,\cdot j} \quad = \quad \mathcal{A}^{-1}\mathcal{H}_j\Sigma \quad \text{pour } j = 0, m. \tag{31}$$

On peut ensuite écrire le système d'équations (12) sous la forme matricielle :

$$C\sigma_{yT,i\cdot} + \Delta T_1 \sigma_{yT,i0} f_1 + \Delta T_{m-1}\sigma_{yT,im} f_{m-1}$$
$$= 3D\sigma_{y,i\cdot} - 3\frac{\Delta T_1}{\Delta T_0}\sigma_{y,i0} f_1 + 3\frac{\Delta T_{m-2}}{\Delta T_{m-1}}\sigma_{y,im} f_{m-1} \quad \text{pour } i = 0, ..., n$$

où e_1^{m-1}, e_{m-1}^{m-1}, C et D sont définies comme précédemment (cf. section B). On connaît maintenant les valeurs de la dérivée aux bords de la grille :

$$\sigma_{yT,ij} = (\mathcal{A}^{-1}\mathcal{H}_j\Sigma)_i = (\mathcal{A}^{-1}\mathcal{H}_j)_{i\cdot} \cdot \Sigma \quad \text{pour } j = 0, m$$

où $(\mathcal{A}^{-1}\mathcal{H}_j)_{i\cdot}$ représente la $i^{\text{ème}}$ ligne de la matrice $(\mathcal{A}^{-1}\mathcal{H}_j)$. On pose

$$P_i \quad = \quad \begin{pmatrix} \boxed{(\mathcal{A}^{-1}\mathcal{H}_0)_{i\cdot}} \\ 0 \quad \dots \quad 0 \\ \vdots \qquad\quad \vdots \\ 0 \quad \dots \quad 0 \\ \boxed{(\mathcal{A}^{-1}\mathcal{H}_m)_{i\cdot}} \end{pmatrix} \quad \in \mathcal{M}_{m-1,n_\Sigma}$$

et

$$\mathcal{G} \quad = \quad \begin{pmatrix} -3\frac{\Delta T_1}{\Delta T_0} & & 0 \\ 0 & \boxed{3D} & \vdots \\ \vdots & & 0 \\ 0 & & 3\frac{\Delta T_{m-2}}{\Delta T_{m-1}} \end{pmatrix} \quad \in \mathcal{M}_{m-1,m+1}$$

Alors

$$\begin{aligned} C\bar{\sigma}_{yT,i\cdot} + P_i\Sigma &= \mathcal{G}\sigma_{y,i\cdot} \\ &= \mathcal{G}L_i\Sigma \end{aligned}$$

où L_i est définie par

$$j^{\text{ème}} \text{ ligne de } L_i \quad = \quad i^{\text{ème}} \text{ ligne de } \mathcal{A}^{-1}\mathcal{B}_j$$

et

$$\bar{\sigma}_{yT,i\cdot} \quad = \quad \begin{pmatrix} \sigma_{yT,i1} \\ \vdots \\ \sigma_{yT,i,M-1} \end{pmatrix}.$$

Remarque 7 *Rappelons que* $n_\Sigma = (n+1)(m+1)+2(n+1)+2(m+1)+4 = (n+3)(m+3)$ *est la taille du vecteur de degrés de liberté* Σ.

On obtient finalement l'expression de $\bar{\sigma}_{yT,i\cdot}$ en fonction de Σ:

$$\bar{\sigma}_{yT,i\cdot} \quad = \quad C^{-1}(\mathcal{G}L_i - P_i)\cdot\Sigma. \tag{32}$$

On peut à présent construire R. Soit

$$R \quad = \quad \begin{pmatrix} \boxed{R_1} \\ \boxed{R_2} \\ \boxed{R_3} \\ \boxed{R_4} \end{pmatrix}$$

où chaque R_i est une matrice de taille $(n+1)(m+1) \times n_\Sigma$ exprimant la dépendance des valeurs de σ et de ses dérivées en chaque point de la grille par rapport au vecteur de paramètres Σ.

1. La construction de R_1 est immédiate puisque les σ_{ij} sont des éléments de Σ:

$$R_1 \quad = \quad \begin{pmatrix} \boxed{I_{(n+1)(m+1)}} & \begin{matrix} 0 & \dots & 0 \\ \vdots & & \vdots \\ 0 & \dots & 0 \end{matrix} \end{pmatrix}$$

2. On peut décomposer R_2 de la façon suivante :

$$R_2 = \begin{pmatrix} R_2^0 \\ \vdots \\ R_2^m \end{pmatrix} \quad \text{avec } R_2^j = \mathcal{A}^{-1}\mathcal{B}_j.$$

3. De la même façon, on construit R_3 :

$$R_3 \quad = \quad \begin{pmatrix} R_3^0 \\ \vdots \\ R_3^M \end{pmatrix}$$

où chaque matrice R_3^j est de taille $(n+1) \times n_\Sigma$. D'après (31), R_3^j est définie par :

$$i^{\text{ème}} \text{ ligne de } R_3^j \quad = \quad j^{\text{ème}} \text{ ligne de } \mathcal{C}^{-1}\mathcal{D}_i.$$

4. Enfin, on écrit R_4 de la manière suivante:

$$R_4 \quad = \quad \begin{pmatrix} R_4^0 \\ \vdots \\ R_4^m \end{pmatrix}$$

où chaque matrice R_4^j est de taille $(n+1) \times n_\Sigma$. D'après (31),

$$\begin{aligned} R_4^0 &= \mathcal{A}^{-1}\mathcal{H}_0, \\ R_4^m &= \mathcal{A}^{-1}\mathcal{H}_m, \end{aligned}$$

et d'après (32), pour $j = 1, ..., m-1$,

$$i^{\text{ème}} \text{ ligne de } R_4^j \quad = \quad j^{\text{ème}} \text{ ligne de } \mathcal{C}^{-1}(\mathcal{G}L_i - P_i).$$

Récapitulatif des principales étapes du calcul de ∇G

$$\boxed{R} \overset{(30)}{\rightarrow} \boxed{\frac{\partial \sigma(y_i, T_j)}{\partial \Sigma}} \overset{(29)}{\rightarrow} \boxed{\frac{\partial \hat{A}^j u^j}{\partial \Sigma}} \overset{(27),(28)}{\rightarrow} \boxed{\begin{array}{c} \dfrac{\partial b^j}{\partial \Sigma} \\ \dfrac{\partial H^j u^{j+1}}{\partial \Sigma} \end{array}} \overset{(26)}{\rightarrow} \boxed{\frac{\partial G}{\partial \Sigma}} \rightarrow \boxed{\nabla_\Sigma G}$$

Références

[1] M. Avellaneda, C. Friedman, R. Holmes, and D. Samperi. Calibrating volatility surfaces via relative entropy minimization. *Applied Mathematical Finance*, (4), 1997.

[2] V. Barette. Un outil en C d'optimisation différentiable sans contraintes. Rapport de stage, Inria, Rocquencourt, France, 2002.

[3] F. Black and M. Scholes. The pricing of options and corporate liabilities. *Journal of Political Economy*, 81:637–654, 1973.

[4] J.F. Bonnans, J.C. Gilbert, C. Lemaréchal, and C. Sagastizábal. *Numerical Optimization: Theoretical and Practical Aspects.* Springer-Verlag, Heidelberg, 2002.

[5] C. Bunks, F.M. Saleck, S. Zaleski, and G. Chavent. Multiscale seismic waveform inversion. *Geophysics*, 60:1457–1473, 1995.

[6] R.H. Byrd, P. Lu, and J. Nocedal. A limited memory algorithm for bound constrained optimization. *SIAM Journal on Scientific and Statistical Computing*, 16:1190–1208, 1995.

[7] F. Clément. MBTT inversion with a poor initial velocity background: Optimization strategies? density of shots? Rapport de recherche 3657, Inria, Rocquencourt, France, 1999.

[8] T.F. Coleman, Y. Li, and A. Verma. Reconstructing the unknown local volatility function. *Journal of Computational Finance*, 2:77–102, 1999.

[9] C. De Boor. Bicubic spline interpolation. *J. Math. Phys.*, 1962.

[10] B. Dupire. Pricing with a smile. *Risk magazine*, 7:18–20, 1994.

[11] N. Jackson, E. Süli, and S. Howison. Computation of deterministic volatility surfaces. *Journal of Computational Finance*, 2:5–32, 1998.

[12] R. Lagnado and S. Osher. A technique for calibrating derivative security pricing models: numerical solution of an inverse problem. *J. Comp. Fin.*, 1:13–25, 1997.

[13] D. Lamberton and B. Lapeyre. *Introduction au calcul stochastique appliqué à la finance.* Ellipses, Paris, 1991.

[14] C. Zhu, R.H. Byrd, and J. Nocedal. L-bfgs-b: Algorithm 778: L-bfgs-b, fortran routines for large scale bound constrained optimization. *ACM Transactions on Mathematical Software*, 23:550–560, 1997.

Unité de recherche INRIA Rocquencourt
Domaine de Voluceau - Rocquencourt - BP 105 - 78153 Le Chesnay Cedex (France)

Unité de recherche INRIA Futurs : Domaine de Voluceau - Rocquencourt - BP 105 - 78153 Le Chesnay Cedex (France)
Unité de recherche INRIA Lorraine : LORIA, Technopôle de Nancy-Brabois - Campus scientifique
615, rue du Jardin Botanique - BP 101 - 54602 Villers-lès-Nancy Cedex (France)
Unité de recherche INRIA Rennes : IRISA, Campus universitaire de Beaulieu - 35042 Rennes Cedex (France)
Unité de recherche INRIA Rhône-Alpes : 655, avenue de l'Europe - 38330 Montbonnot-St-Martin (France)
Unité de recherche INRIA Sophia Antipolis : 2004, route des Lucioles - BP 93 - 06902 Sophia Antipolis Cedex (France)

Éditeur
INRIA - Domaine de Voluceau - Rocquencourt, BP 105 - 78153 Le Chesnay Cedex (France)
http://www.inria.fr
ISSN 0249-6399